NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

FREQUENCY DOMAIN STRUCTURAL IDENTIFICATION

by

Richard Johnson

June 1996

Thesis Advisor: Joshua H. Gordis

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

blic reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and intaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, cluding suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

TITLE AND SUBTITLE FREQUENCY DOMAIN STRUCTURAL IDENTIFICATION	5. FUNDING NUMBERS
AUTHOR(S) Richard Johnson	
PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) aval Postgraduate School Ionterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official olicy or position of the Department of Defense or the U.S. Government.

a. DISTRIBUTION/AVAILABILITY STATEMENT pproved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

. ABSTRACT *(maximum 200 words)*

The Structural Synthesis Transformation is used to conduct structural system identification in the frequency domain. For spatially complete ses where each of the frequency response functions at every degree of freedom of each of the coordinates of the modeled system are available it shown that the theory exactly identifies all modeling errors. For spatially incomplete cases where the frequency response functions are ailable only at a proper subset of the degrees of freedom of the finite element model, single mode solutions are computed over intervals about e modes of the experimental system using matrices and complex valued line integrals. Methods of forming multiple mode solutions from the ngle mode solutions are explored.

. SUBJECT TERMS Finite Element Method	15. NUMBER OF PAGES 161
	16. PRICE CODE

. SECURITY CLASSIFICATION OF EPORT nclassified	18. SECURITY CLASSIFI-CATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFI-CATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-28-5500 Standard Form 298 (Rev. 289)

Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

FREQUENCY DOMAIN STRUCTURAL IDENTIFICATION

Richard Johnson
Lieutenant Commander, United States Navy
B.S., Southern University, 1974
M.S., Louisiana State University, 1978

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
from the
NAVAL POSTGRADUATE SCHOOL
June 1996

Author: ______________________
Richard Johnson

Approved by: ______________________
Joshua H. Gordis, Thesis Advisor

____
Terry R. McNelley, Chairman
Department of Mechanical Engineering

ABSTRACT

The Structural Synthesis Transformation is used to conduct structural system identification in the frequency domain. For spatially complete cases where each of the frequency response functions at every degree of freedom of each of the coordinates of the modeled system are available it is shown that the theory exactly identifies all modeling errors. For spatially incomplete cases where the frequency response functions are available only at a proper subset of the degrees of freedom of the finite element model, single mode solutions are computed over intervals about the modes of the experimental system using matrices and complex valued line integrals. Methods of forming multiple mode solutions from the single mode solutions are explored.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The Finite Element (FE) method is a proven tool for modeling structural dynamic systems. As the complexity of the system increases the FE model may not accurately reflect the dynamic behavior of the system. To determine the extent to which the FE model accurately describes the physical system, a comparison of the dynamic response or modal parameters of the system as predicted by the FE model and the response or modal parameters of the physical system as determined by measurements of the dynamic response of the system can be made. Such a comparison can easily point out the differences in the dynamic behavior of the FE model and the physical system but fail to provide the necessary corrections to the FE model that will provide a more accurate FE model prediction of the dynamic behavior of the physical system.

Structural system identification refers to procedures used to identify finite element modeling errors using dynamic test data. Localization is the process of identifying those degrees of freedom (DOF) of the FE model whose impedance differs from that of the physical system. We refer to this set of DOF as the error set. Identification is the process of finding matrices ΔK, ΔM, and ΔC that are corrections to the FE model stiffness, mass, and damping matrices. When corrections are installed, the frequency response of the corrected FE model better predicts the frequency response of the experimental system, and hence the modal parameters of the corrected FE model better predicts those of the experimental system.

Structural system identification can be conducted using either modal or frequency domain methods. This thesis investigates a frequency domain method based on the structural synthesis transformation (SST) as outlined in Reference (1). When measured dynamic response data is available for each DOF of each coordinate of the FE model the identification is termed *spatially complete* and the SST can be used to exactly determine which elements of the FE model are in error. Additionally, the SST can be used to compute correction matrices ΔM, ΔK, and ΔC that can be used to correct the mass, stiffness, and damping matrices of the FE model. The dynamic response of the corrected

FE model exactly matches that of the physical system. In the more frequent case that dynamic response data is not available at every DOF of the FE model, the identification is termed *spatially incomplete*. In this case the SST provides a frequency dependent solution.

SST-based structural system identification uses the frequency response function (FRF) of the structure under consideration. The FRF is a quantitative description of the dynamic behavior of the structure. To obtain the FRF, known harmonic excitation forces are applied and the resulting harmonic response of the structure measured. The ratio of excitation forces to response at a coordinate evaluated at each frequency in a specified bandwidth defines the FRF.

II. THEORY

A. IMPEDANCE DESCRIPTION

The impedance model of a given physical system can be defined by the relationship of structural displacement with respect to an applied force,

$$\begin{Bmatrix} f_i \\ f_c \end{Bmatrix} = \begin{bmatrix} Z^a_{ii} & Z^a_{ic} \\ Z^a_{ic} & Z^a_{cc} \end{bmatrix} \begin{Bmatrix} x_i \\ x_c \end{Bmatrix} \tag{2.1a}$$

The harmonic force and response vectors are denoted by "f" and "x" respectively. These vectors and the impedance matrix, Z, are in general complex-valued and frequency dependent. Subscripts "i" and "c" denote non-error and error DOF respectively. The superscript "a" denotes quantities calculated from a FE (analytic) model. If the values were obtained from experimental test data, the superscript would be "x". Thus, for the experimental model the impedance relationship would be given by:

$$\begin{Bmatrix} f_i \\ f_c \end{Bmatrix} = \begin{bmatrix} Z^x_{ii} & Z^x_{ic} \\ Z^x_{ic} & Z^x_{cc} \end{bmatrix} \begin{Bmatrix} x_i \\ x_c \end{Bmatrix} \tag{2.1b}$$

In pratice, the elements of the experimental impedance matrix of Equation (2.1b) are unmeasurable quanties. To see this we expand row j of Equation (2.1b) to obtain

$$f_j = z^x_{j1}x_1 + z^x_{j2}x_2 + \cdots + z^x_{jn}x_n \tag{2.2}$$

To measure an element z^x_{j1} of Z^a, requires that we impose a unit displacement at coordinate x_1 while physically holding all other other coordinates at zero displacements. This is not physically possible. Assuming, for purpose of definition, the availability of the experimental impedance matrix, the quantitative difference between the analytical and experimental systems, as a function of frequency, is described by the impedance error

matrix. It is defined by the difference between the analytical and experimental impedance matrices. The error impedance matrix relationship is defined as:

$$\begin{bmatrix} 0 & 0 \\ 0 & \Delta Z(\Omega) \end{bmatrix} = \begin{bmatrix} Z_{ii}^a & Z_{ic}^a \\ Z_{ci}^a & Z_{cc}^a \end{bmatrix} - \begin{bmatrix} Z_{ii}^x & Z_{ic}^x \\ Z_{ci}^x & Z_{cc}^x \end{bmatrix} \tag{2.3}$$

B. STRUCTURAL SYNTHESIS TRANSFORMATION

Since the experimental impedance matrix, Z^x, is in general unavailable, frequency domain structural synthesis is used to identify the impedance error matrix using FRF data exclusively. A structural synthesis transformation is constructed from ΔZ of Equation (2.3) which encompasses the FE model errors. This transformation is applied to the finite element model to produce an experimental system FRF.

The FRF relates structural response to applied excitation. Given a FE model with impedance matrix, Z^a, the FRF, H^a, is the matrix inverse of the impedance matrix Z^a of equation (2.1a). For a static system (Ω=0) the FRF is simply the flexibility matrix (inverse stiffness matrix). Using the notation of Equation (2.1) we may partition H^a as follows:

$$\begin{Bmatrix} x_i \\ x_c \end{Bmatrix} = \begin{bmatrix} H_{ii}^a & H_{ic}^a \\ H_{ic}^a & H_{cc}^a \end{bmatrix} \begin{Bmatrix} f_i \\ f_c \end{Bmatrix} \tag{2.4}$$

Generally, the "c" response coordinates experience applied forces due to both error impedances and externally applied forces, whereas "i" response coordinates experience only externally applied forces, such that,

$$f_c = f_c^{ext} + f_c^{\Delta Z} \tag{2.5a}$$

and

$$f_i = f_i^{ext} \tag{2.5b}$$

Expanding Equation (2.4) and substituting the relations of Equation (2.5) yeilds the following relationships:

$$x_i = H^a_{ii} f^{ext}_i + H^a_{ic} f^{ext}_c + H^a_{ic} f^{\Delta Z}_c \tag{2.6a}$$

$$x_c = H^a_{ci} f^{ext}_i + H^a_{cc} f^{ext}_c + H^a_{cc} f^{\Delta Z}_c \tag{2.6b}$$

If we include a copy of Equation (2.6b), in expanded matrix notation, Equation (2.6) reflects the three harmonic excitation terms to be considered, i.e.,

$$\begin{Bmatrix} x_x \\ x_c \\ x_c \end{Bmatrix} = \begin{bmatrix} H^a_{ii} & H^a_{ic} & H^a_{ic} \\ H^a_{ci} & H^a_{cc} & H^a_{cc} \\ H^a_{ci} & H^a_{cc} & H^a_{cc} \end{bmatrix} \begin{Bmatrix} f^{ext}_i \\ f^{ext}_c \\ f^{\Delta Z}_c \end{Bmatrix} \tag{2.7}$$

Response coordinates "c" and "i", which are due to external forces, will hereafter be referred to as "e" coordinates, denoting their dependence on external force excitation. Consequently, the three excitation forces are condensed into two under the identity:

$$\{f_e\} = \left[\left[f^{ext}_i\right] \quad \left[f^{ext}_c\right]\right]^T \tag{2.8a}$$

$$\{f_c\} = \left\{f^{\Delta Z}_c\right\} \tag{2.8b}$$

and Equation (2.7) reduces to

$$\begin{Bmatrix} x_e \\ x_c \end{Bmatrix} = \begin{bmatrix} H^a_{ee} & H^a_{ec} \\ H^a_{ce} & H^a_{cc} \end{bmatrix} \begin{Bmatrix} f_e \\ f_c \end{Bmatrix} \tag{2.9}$$

Equation (2.3) shows that the impedance error is defined as the difference between the analytic and experimental impedance models. Hence, a transformation is required which uses the FRF relationship of Equation (2.9) to generate a similar relationship for the

experimental system. The impedance error ΔZ provides the basis by which this transformation is developed.

The impedance error matrix, $\Delta Z(\Omega)$, is a function of frequency and satisfies:

$$\{f_c\} = -[\Delta Z(\Omega)]\{x_c\} \tag{2.10}$$

where

$$[\Delta Z(\Omega)] = [[\Delta K] + j\Omega[\Delta C] - \Omega^2[\Delta M]] \tag{2.11}$$

for Ω the forcing frequency, $j = \sqrt{-1}$ and ΔK, ΔC, and ΔM stiffness, damping and mass error matrices comparable to those of the finite element formulation. The minus sign in Equation (2.10) reflects that the reaction forces imposed by impedance errors on the baseline model are being considered. Substituting the relationship

$$\begin{Bmatrix} f_e \\ f_c \end{Bmatrix} = \begin{bmatrix} I & 0 \\ 0 & -\Delta Z_c \end{bmatrix} \begin{Bmatrix} f_e \\ x_c \end{Bmatrix} \tag{2.12}$$

into Equation (2.9) yields:

$$\begin{Bmatrix} x_e \\ x_c \end{Bmatrix}^* = \begin{bmatrix} H^a_{ee} & H^a_{ec} \\ H^a_{ce} & H^a_{cc} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -\Delta Z \end{bmatrix} \begin{Bmatrix} f_e \\ x_c \end{Bmatrix}^* \tag{2.13}$$

simplifying we get:

$$\begin{Bmatrix} x_e \\ x_c \end{Bmatrix}^* = \begin{bmatrix} H^a_{ee} & -H^a_{ec}\Delta Z \\ H^a_{ce} & -H^a_{cc}\Delta Z \end{bmatrix} \begin{Bmatrix} f_e \\ x_c \end{Bmatrix}^* \tag{2.14}$$

Expanding Equation (2.14) into two equations and using "*" to denote a synthesized modified response results in

$$x^*_c = H^a_{ce} f_e - H^a_{cc}\Delta Z x^*_c \tag{2.15a}$$

$$x^*_e = H^a_{ee} f_e - H^a_{ec}\Delta Z x^*_c \tag{2.15b}$$

Rearranging Equation (2.15b) produces

$$\left[I + H_{cc}^{a}\Delta Z\right]x_{c}^{*} = H_{ce}^{a}f_{e} \tag{2.16a}$$

Using the property of the frequency response function,

$$x_{c} = H_{ce}^{a}f_{e} \tag{2.16b}$$

$$\left[I + H_{cc}^{a}\Delta Z\right]x_{c}^{*} = x_{c} \tag{2.16c}$$

$$x_{c}^{*} = \left[I + H_{cc}^{a}\Delta Z\right]^{-1}x_{c} \tag{2.16d}$$

Introducing Equation (2.16) into Equation (2.15b) results in

$$x_{c}^{*} = H_{ee}^{a}f_{e} - H_{ec}^{a}\Delta Z\left[I + H_{cc}^{a}\Delta Z\right]^{-1}x_{c} \tag{2.17a}$$

$$x_{c}^{*} = H_{ee}^{a}f_{e} - H_{ec}^{a}\Delta Z\left[I + H_{cc}^{a}\Delta Z\right]^{-1}H_{ce}^{a}f_{e} \tag{2.17b}$$

Once again we recall the property of a FRF,

$$x_{e}^{*} = H_{ee}^{*}f_{e} \tag{2.18a}$$

Combining Equations (2.17b) and (2.18a) yields

$$H_{ee}^{*} = H_{ee}^{a} - H_{ec}^{a}\Delta Z\left[I + H_{cc}^{a}\Delta Z\right]^{-1}H_{ce}^{a} \tag{2.18b}$$

Noting that

$$\left[I + H_{cc}^{a}\Delta Z\right]^{-1} = \left[\left(\Delta Z^{-1} + H_{cc}^{a}\right)\Delta Z\right]^{-1} \tag{2.19a}$$

and applying the matrix property

$$\left([a][b]\right)^{-1} = [b]^{-1}[a]^{-1} \tag{2.19b}$$

we get that

$$H_{ee}^{*} = H_{ee}^{a} - H_{ec}^{a}\left[\Delta Z^{-1} + H_{cc}^{a}\right]^{-1}H_{ce}^{a} \tag{2.20a}$$

Replacing the superscript "*", which denotes the structures's synthesized coupled response, with the superscript "x" to indicate the test system response we arrive at

$$H_{ee}^{x} = H_{ee}^{a} - H_{ec}^{a}\left[\Delta Z^{-1} + H_{cc}^{a}\right]^{-1} H_{ce}^{a} \tag{2.20b}$$

In full matrix notation we have

$$\begin{bmatrix} H_{ii}^{x} & H_{ic}^{x} \\ H_{ci}^{x} & H_{cc}^{x} \end{bmatrix} = \begin{bmatrix} H_{ii}^{a} & H_{ic}^{a} \\ H_{ci}^{a} & H_{cc}^{a} \end{bmatrix} - \begin{bmatrix} H_{ic}^{a} \\ H_{cc}^{a} \end{bmatrix} \left[\Delta Z^{-1} + H_{cc}^{a}\right]^{-1} \begin{bmatrix} H_{ic}^{a} \\ H_{cc}^{a} \end{bmatrix}^{T} \tag{2.20c}$$

Equation (2.20b) is the structural synthesis transformation equation. When the experimental system FRF, H^{x} , is available the SST can be used to identify a frequency dependent impedance error matrix [ΔZ(Ω)]. Additionally, using Equation (2.9), [ΔZ(Ω)] can be decomposed into constitutent stiffness, mass, and damping errors.

C. FREQUENCY DOMAIN LOCALIZATION

We may rewrite Equation (2.20b) as

$$\Delta H_{ee} = H_{ec}^{a} D^{-1} H_{ce}^{a} \tag{2.21}$$

where

$$\Delta H_{ee} = H_{ee}^{a} - H_{ee}^{x} \tag{2.22a}$$

and

$$D = \left[\Delta Z^{-1} + H_{ce}^{a}\right] \tag{2.22b}$$

We define the localization matrix L as

$$L = Z_{ee}^{a} \cdot \Delta H_{ee} \cdot Z_{ee}^{a} \tag{2.23}$$

using the expression of Equation (2.22a) in Equation (2.23) we can rewrite L as

$$L = Z_{ee}^{a} \cdot H_{ec}^{a} D^{-1} H_{ce}^{a} \cdot Z_{ee}^{a} \tag{2.24a}$$

Expanding the "e" coordinate set into error and non error coordinates we get

$$L=\begin{bmatrix} Z_{ii}^a & Z_{ic}^a \\ Z_{ci}^a & Z_{cc}^a \end{bmatrix}\begin{bmatrix} H_{ic}^a \\ H_{cc}^a \end{bmatrix}\left[D^{-1}\right]\left[H_{ic}^a \quad H_{cc}^a\right]\begin{bmatrix} Z_{ii}^a & Z_{ic}^a \\ Z_{ci}^a & Z_{cc}^a \end{bmatrix} \tag{2.24b}$$

Noting the the frequency response matrix is the inverse of the impedance matrix, i.e.,

$$\begin{bmatrix} Z_{ii}^a & Z_{ic}^a \\ Z_{ci}^a & Z_{cc}^a \end{bmatrix}\begin{bmatrix} H_{ii}^a & H_{ic}^a \\ H_{ci}^a & H_{cc}^a \end{bmatrix}=\begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{2.25a}$$

all mixed product coordinates of Equation (2.24b) must be zero and L simplifies to

$$L=\begin{bmatrix} 0 & 0 \\ 0 & D^{-1} \end{bmatrix} \quad @\ \Omega=\Omega_i \tag{2.25b}$$

D. ERROR IMPEDANCE

We can solve Equation (2.20b) for the impedance error *[ΔZ]*. From Equation (2.25b) terms of Equation (2.20b) associated with non error coordinates may be assumed to be zero. We get the following form of the impedance error matrix

$$[\Delta Z]=\left(\left[\tilde{H}_{cc}^x\right]-\left[H_{cc}^a\right]\right)^{-1} \quad @\ \Omega=\Omega_i \tag{2.26a}$$

where

$$\left[\tilde{H}_{cc}^x\right]=\left(\left[H_{cc}^a\right]^{-1}[\Delta H]\left[H_{cc}^a\right]^{-1}\right)^{-1} \tag{2.26b}$$

Let $\Xi=\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$ be a set of frequencies where $\Omega_1 < \Omega_2 < \ldots < \Omega_{n-1} < \Omega_n$. If for each i= 2,3, ..., n-1, we apply Equation (2.10) at each of the frequencies Ω_{i-1}, Ω_i , and Ω_{i+1} and assemble the resulting three equations into a system of three equations in three unknowns, we get the matrix equation

$$\begin{bmatrix} \Delta Z_c(\Omega_{i-1}) \\ \Delta Z_c(\Omega_i) \\ \Delta Z_c(\Omega_{i+1}) \end{bmatrix}=\begin{bmatrix} I & -\Omega_{i-1}^2 I & j\Omega_{i-1}I \\ I & -\Omega_i^2 I & j\Omega_i I \\ I & -\Omega_{i+1}^2 I & j\Omega_{i+1}I \end{bmatrix}\begin{bmatrix} \Delta K_c \\ \Delta M_c \\ \Delta C_c \end{bmatrix} \tag{2.27}$$

Equation (2.27) can be used to decompose the frequency dependent impedance error into constitutent stiffness, mass, and damping error matrices, *ΔK*, *ΔM*, and *ΔC* at the frequencies Ω_i, i=2,3,...,n-1. These constitutents matrices are in general frequency

dependent. Denoting the solution of Equation (2.27) by $\begin{Bmatrix} \Delta K_c(\Omega_i) \\ \Delta M_c(\Omega_i) \\ \Delta C_c(\Omega_i) \end{Bmatrix}$ for i=2,3,...,n-1 we obtain for each Ω_i i=2,3,...,n-1, error stiffness, mass, and damping matrices $\Delta K_c(\Omega_i)$, $\Delta M_c(\Omega_i)$, and $\Delta C_c(\Omega_i)$ such that

$$\Delta Z_c(\Omega_i) = \Delta K_c(\Omega_i) - \Omega_i^2 \Delta M_c(\Omega_i) + j\Omega_i \Delta C_c(\Omega_i) \tag{2.28}$$

The matrices $\Delta K_c(\Omega_i)$, $\Delta M_c(\Omega_i)$, and $\Delta C_c(\Omega_i)$ can be used to numerically correct the stiffness, mass, and damping matrices of the FE model at the frequencies Ω_i i=2,3,...,n-1 in that the FRF of the corrected FE model at the frequencies Ω_i approximates the experimental system FRF at Ω_i. To express this symbolically, if K^a, M^a, and C^a are the stiffness, mass, and damping matrices of the FE model and H^x is the FRF matrix of the experimental system at Ω_i

$$H^x(\Omega_i) \approx \left(K^a + \Delta K_c(\Omega_i) - \Omega_i^2\left(M^a + \Delta M_c(\Omega_i)\right) + j\Omega_i\left(C^a + \Delta C_c(\Omega_i)\right)\right)^{-1} \tag{2.29}$$

where the "c" subscripted matrices are added at the corresponding error set coordinates of the "a" superscripted matrices.

For a general set of frequencies, $\Xi=\{\Omega_1, \Omega_2, \ldots, \Omega_n\}$, we can form the system of n equations in three unknowns given by:

$$\begin{bmatrix} \Delta Z_c(\Omega_1) \\ \vdots \\ \Delta Z_c(\Omega_i) \\ \vdots \\ \Delta Z_c(\Omega_n) \end{bmatrix} = \begin{bmatrix} I & -\Omega_1^2 I & j\Omega_1 I \\ \vdots & \vdots & \vdots \\ I & -\Omega_i^2 I & j\Omega_i I \\ \vdots & \vdots & \vdots \\ I & -\Omega_n^2 I & j\Omega_n I \end{bmatrix} \begin{bmatrix} \Delta K_c \\ \Delta M_c \\ \Delta C_c \end{bmatrix} \tag{2.30}$$

The solution $\begin{bmatrix} \Delta K_c \\ \Delta M_c \\ \Delta C_c \end{bmatrix} = \begin{bmatrix} \Delta K_c(\Xi) \\ \Delta M_c(\Xi) \\ \Delta C_c(\Xi) \end{bmatrix}$ of Equation (2.30) represents error stiffness, mass, and damping matrices which best corrects the FE model in a least squares sense. Equations (2.27) and (2.30) are fundamental to all that follows.

III. SPATIALLY COMPLETE STRUCTURAL IDENTIFICATION

To illustrate the principles of SST based frequency domain structural identification we will use the FE model of a free-free beam. To simulate the experimental system we impose a 25% addition to the mass and stiffness of elements 3 and 4 of a 10 element FE model. Figure 3-1 shows the finite element model of the beam and the spatially complete experimental system that results from imposing the mass and stiffness additions at element 3 and 4 of the FE model. Table 3-1 shows the system frequencies of the analytic and experimental systems.

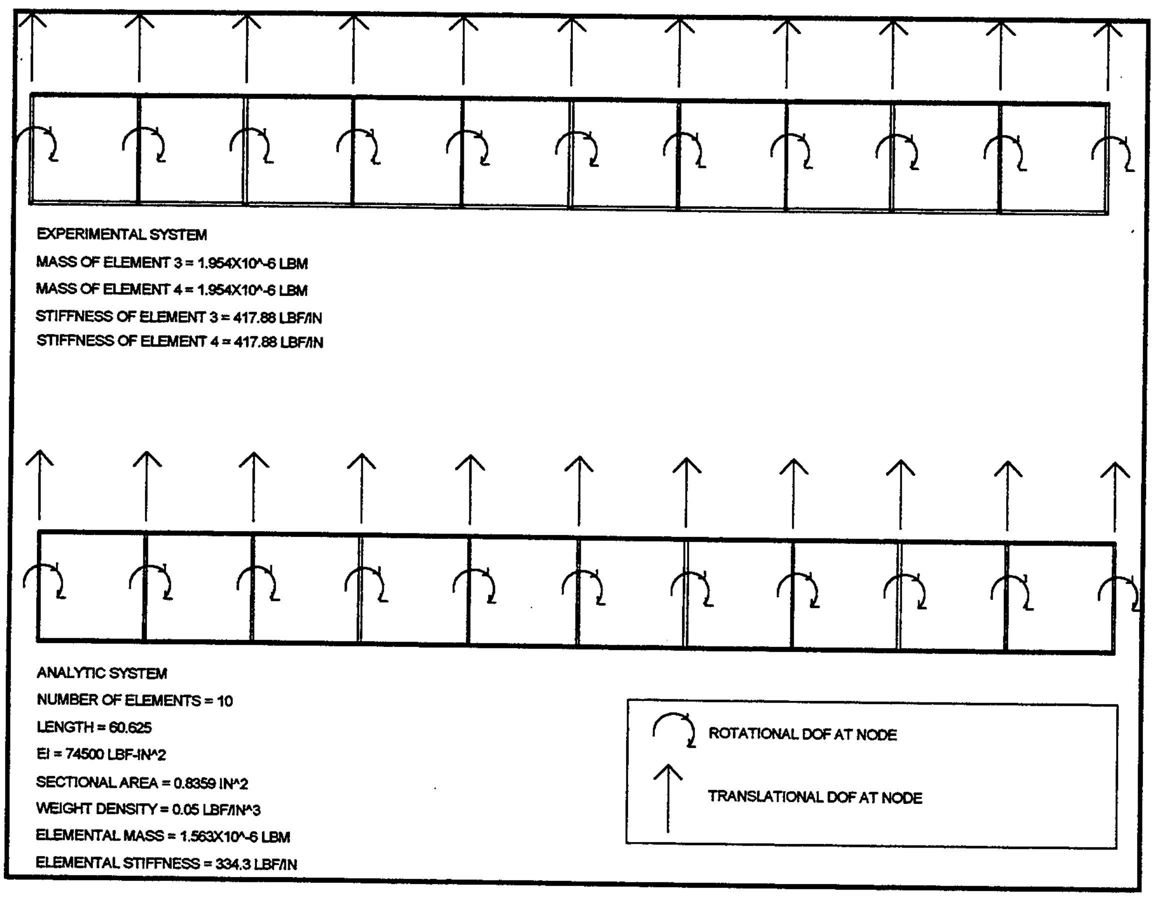

Figure 3- 1 Spatially complete Analytic and Experimental Systems.

MODE	ANALYTIC (Hz)	EXPERIMENTAL (Hz)
1	**25.41**	**24.42**
2	**70.06**	**68.41**
3	**137.4**	**130.2**
4	**227.5**	**216.3**
5	**340.8**	**329.4**
6	**478.0**	**456.5**
7	**639.9**	**612.0**
8	**826.5**	**802.6**
9	**1026**	**969**
10	**1363**	**1323**
11	**1641**	**1573**
12	**1981**	**1920**
13	**2381**	**2283**
14	**2850**	**2754**
15	**3397**	**3269**
16	**4028**	**3912**
17	**4720**	**4605**
18	**5377**	**5156**
19	**6795**	**6791**
20	**6808**	**6801**

Table 3-1 System Frequencies of spatially complete Analytic and Experimental systems.

We will denote by M^a, K^a, and C^a the mass stiffness and damping matrices of the FE model and by M^x, K^x, and C^x the mass stiffness and damping matrices of the experimental system. The impedance matrices of the analytic and simulated experimental systems are given by:

$$Z^a(\Omega) = K^a + j\Omega C^a - \Omega^2 M^a \tag{3.1a}$$

$$Z^x(\Omega) = K^x + j\Omega C^x - \Omega^2 M^x \tag{3.1b}$$

The FRF matrices of the analytic and simulated experimental systems are given by:

$$H^a(\Omega) = Z^a(\Omega)^{-1} \tag{3.2a}$$

$$H^x(\Omega) = Z^x(\Omega)^{-1} \tag{3.2b}$$

Figure 3-2 shows a comparison of the driving point FRF at DOF 1 of the analytic and experimental system.

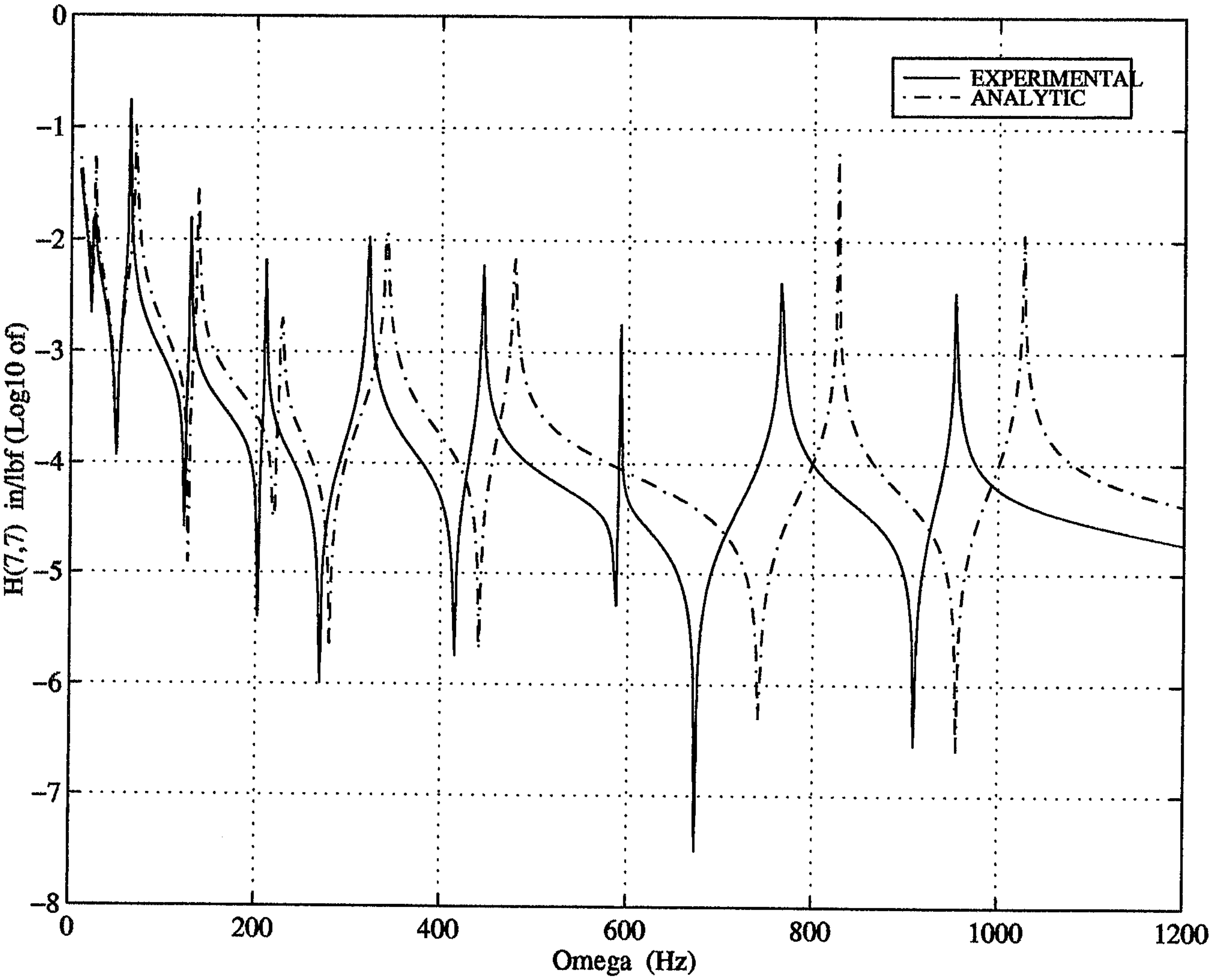

Figure 3- 2 Analytic FRF vs simulated Experimental FRF.

For a given frequency Ω_0, using Equation (2.24a), we can form the localization matrix L. Plotting the diagonal elements of L versus the associated DOF we obtain the plot shown in Figure 3.3. For each frequency, Ω, in a given frequency range, we can compute the diagonal of L at Ω. Assembling the diagonals over the frequency range of our system into a rectangular matrix and performing a MATLAB mesh plot of the resulting rectangular matrix we obtain the surface plot shown in figure 3.4. Figure 3.4 shows the frequency dependency of the localization matrix diagonals. As our system is spatially complete Equation (2.25) forces all non error coordinates to be zero. From Figure 3-3 we can determine that the locations of the nonzero diagonal values are DOF 9, 10, 11, and 12. We denote by C_{err} the set of DOF for which L(i,i) is non zero. For our beam system $C_{err} = \{9,10,11,12\}$. Figure 3.5 shows the frequency dependence of typical error and non error set diagonal elements of the localization matrix L.

Using the set, C_{err}, which results from the localization, we can perform a partitioning of the FRF matrix as described by Equation(2.4). We can now apply Equation (2.26) to compute ΔZ as a matrix function of frequency over the frequency range of our system. We then use Equation (2.27) to decompose ΔZ into its constituent components ΔK, ΔM, and ΔC. We get exact solutions of error stiffness, mass, and damping as shown in figures 3-7, 3-8 and 3-9. The MATLAB Routine SST.M of Appendix A can be used to accomplish the above steps.

For each Ω in the frequency range of our system we can form the sum:

$$Z_{corr}(\Omega) = Z_{cc}^{a}(\Omega) + \Delta Z(\Omega) \tag{3.3}$$

We refer to

$$H_{corr} = \left(Z_{corr}\right)^{-1} \tag{3.4}$$

as the corrected FRF of the analytic model. H_{corr} is the FRF of the model that results from installing the corrections as identified by the Equation (2.26a). Figure 3-10 shows a comparison plot of the FRF of the corrected model and the FRF of the experimental system. Figure 3-10 clearly shows the exactness of the SST solution in the case of a

spatially complete system. The experimental and corrected model FRFs are identical to within plot resolution.

At each frequency, Ω, in the frequency range of our system we can form the sum

$$Z_{corr}^{const}(\Omega) = Z_{cc}^{a}(\Omega) + \Delta K(\Omega) + j\Omega\Delta C(\Omega) - \Omega^2 \Delta M(\Omega) \tag{3.5}$$

We refer to:

$$H_{corr}^{const} = \left(Z_{corr}^{const}\right)^{-1} \tag{3.6}$$

as the constituent corrected FRF. Figure 3-11 is a comparison plot of the constitutent corrected FRF and experimental for our system. The offset between the two plots is exactly equal to the sampling frequency $\Delta\Omega$.

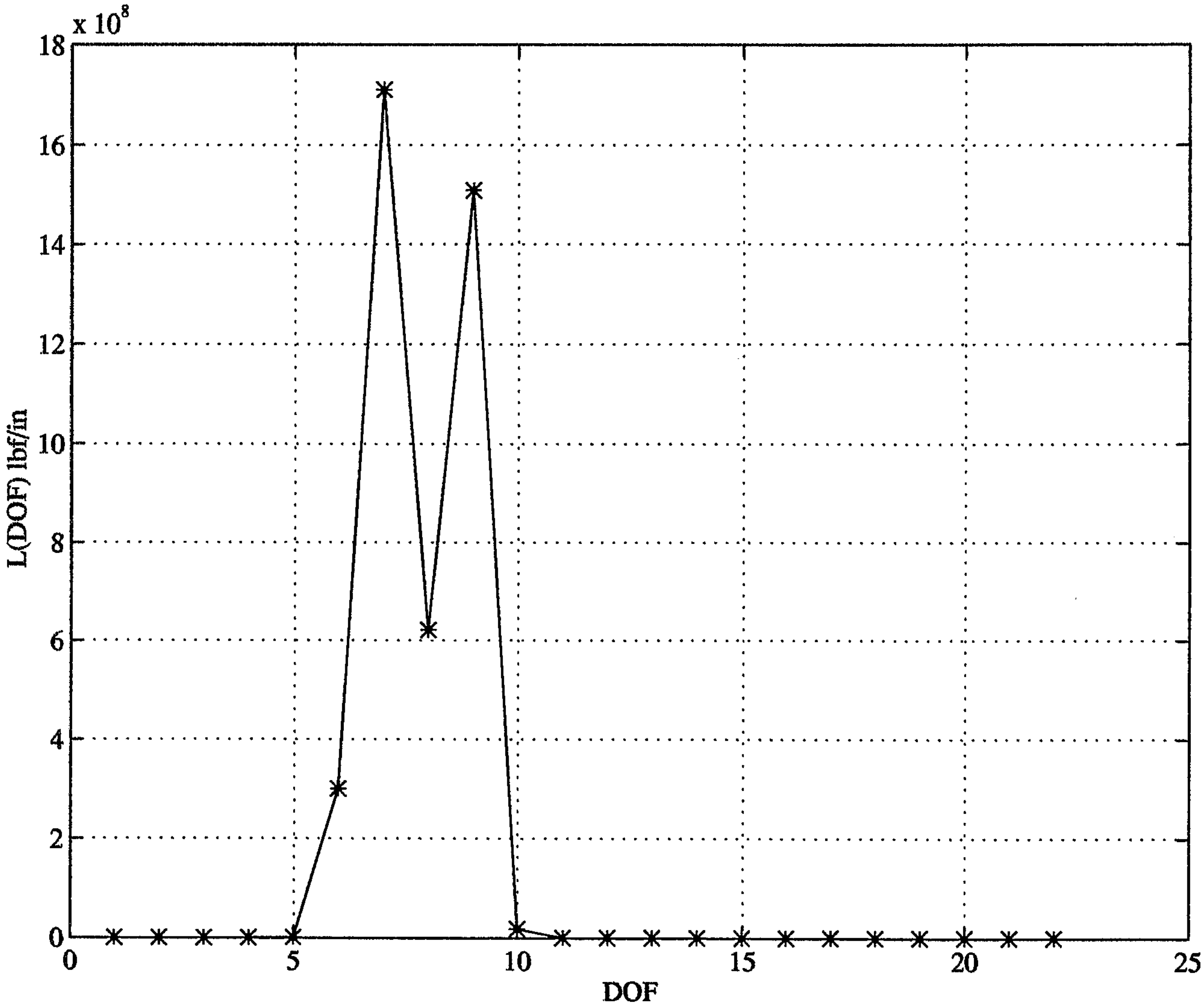

Figure 3- 3 Spatially complete localization matrix diagonal at Ω=352 Hz.

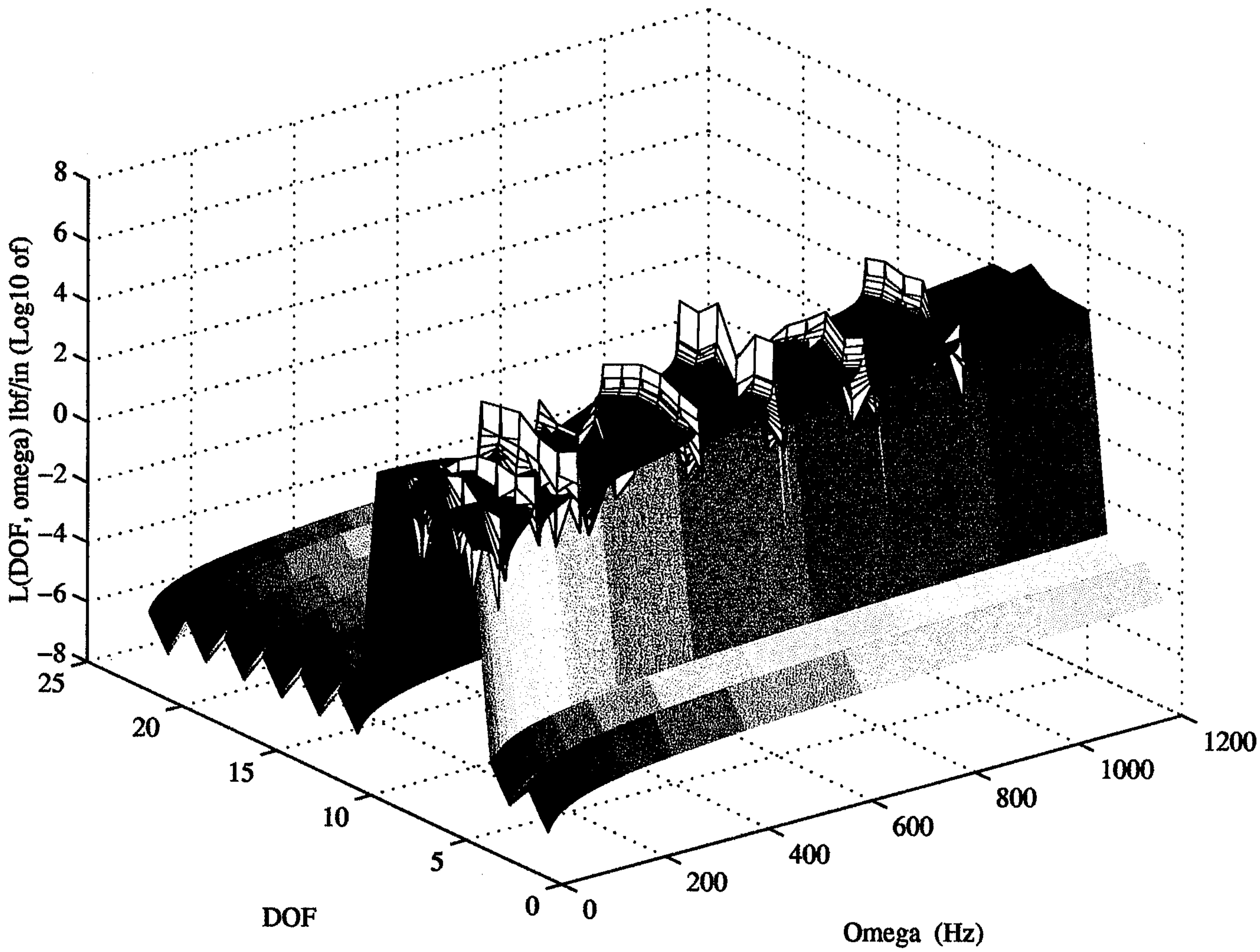

Figure 3- 4 Frequency dependence of spatially complete localization matrix diagonals.

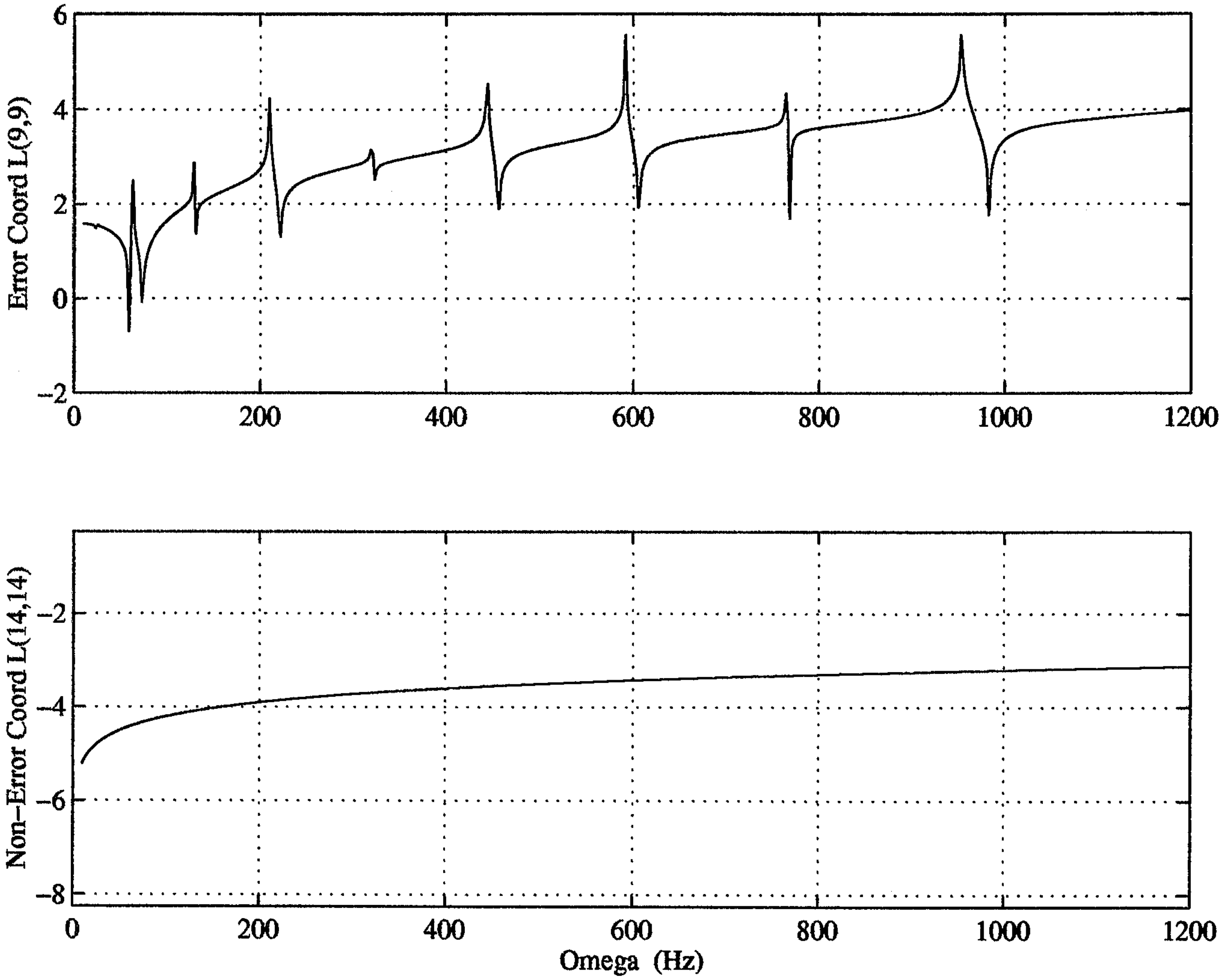

Figure 3- 5 Frequency dependence of spatially complete localization matrix error set DOF. Units are lbf/in (Log10 of).

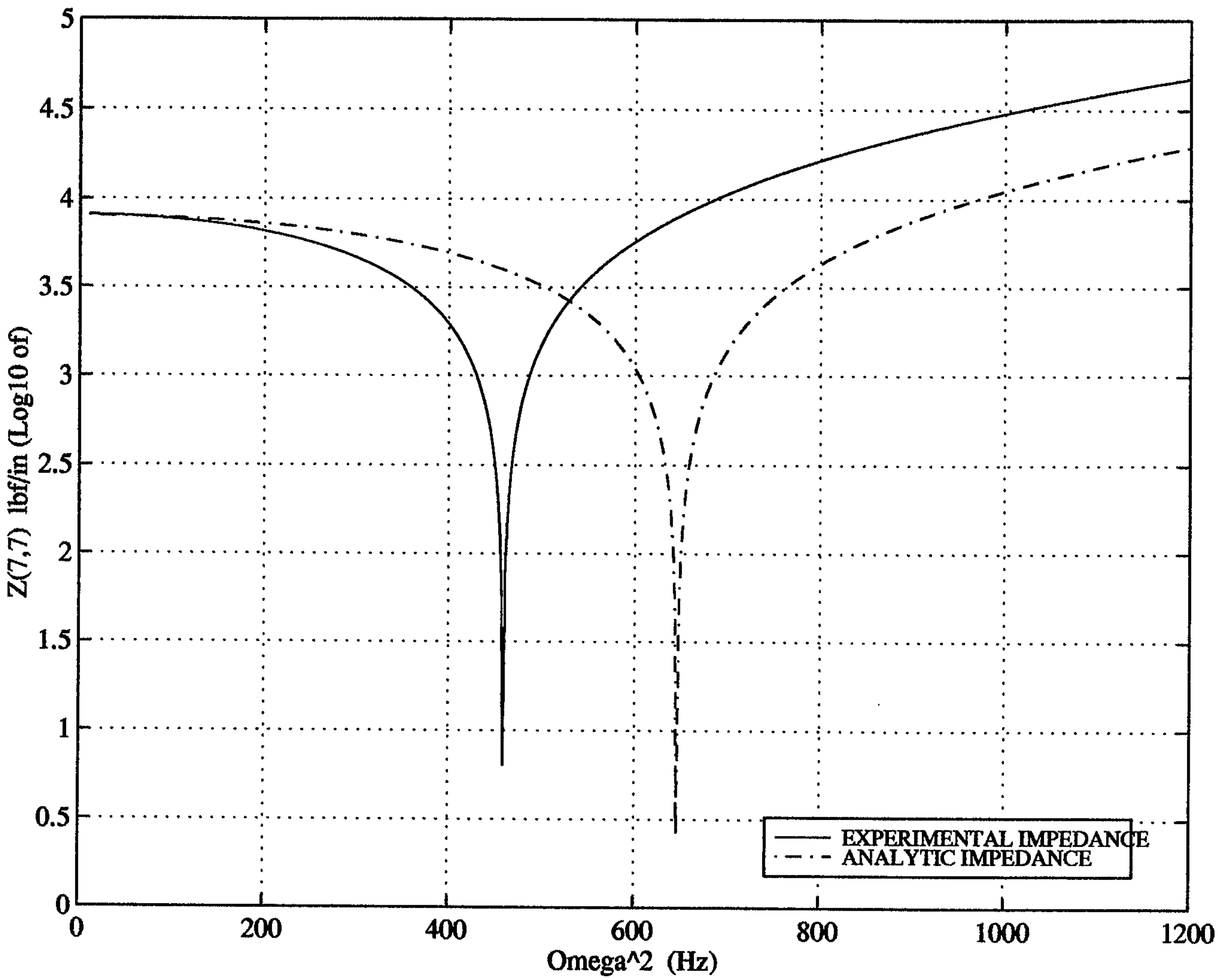

Figure 3- 6 Spatially complete Analytic Impedance vs Experimental impedance.

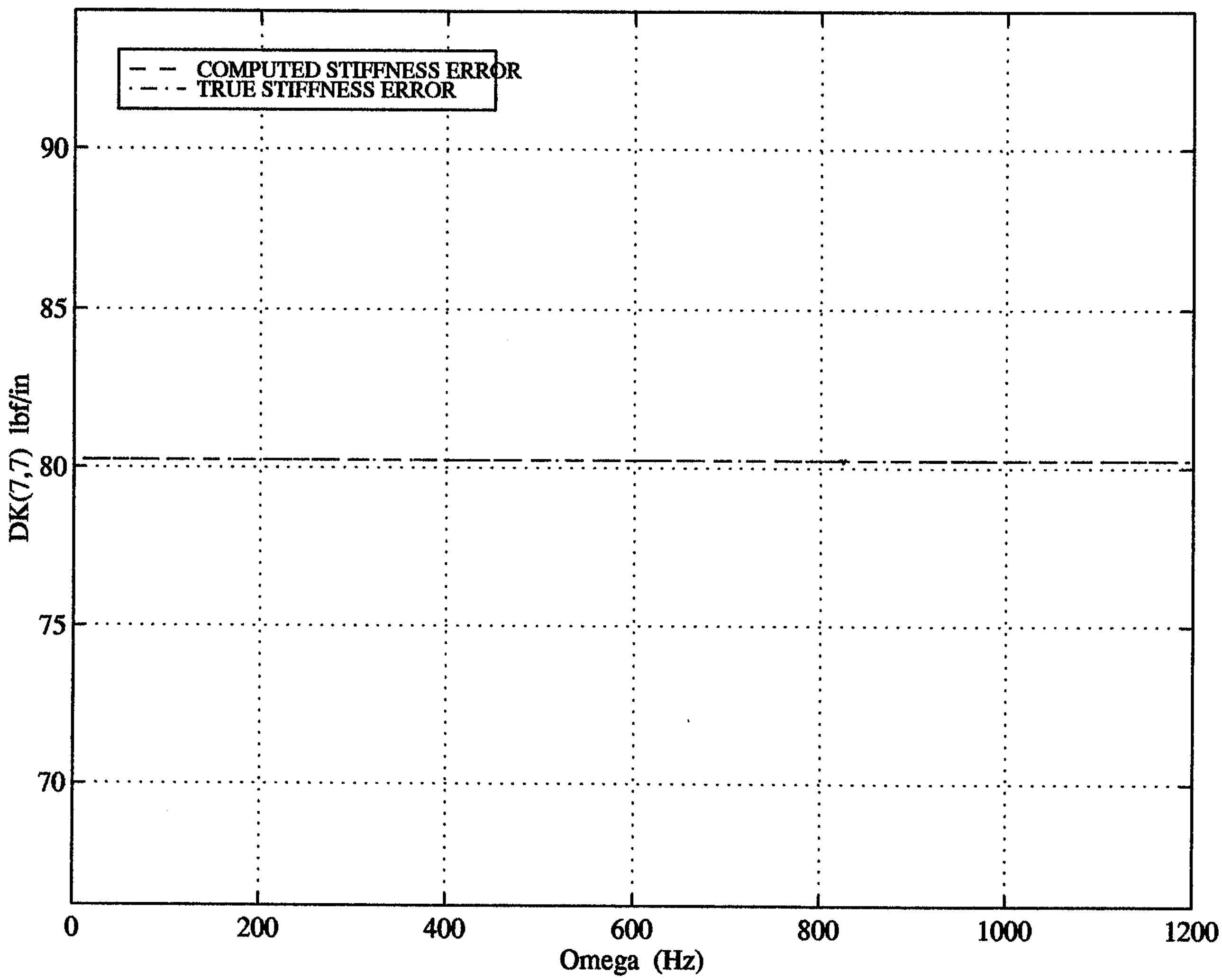

Figure 3- 7 Computed stiffness vs true stiffness for spatially complete beam. Plots are identical within plot resolution.

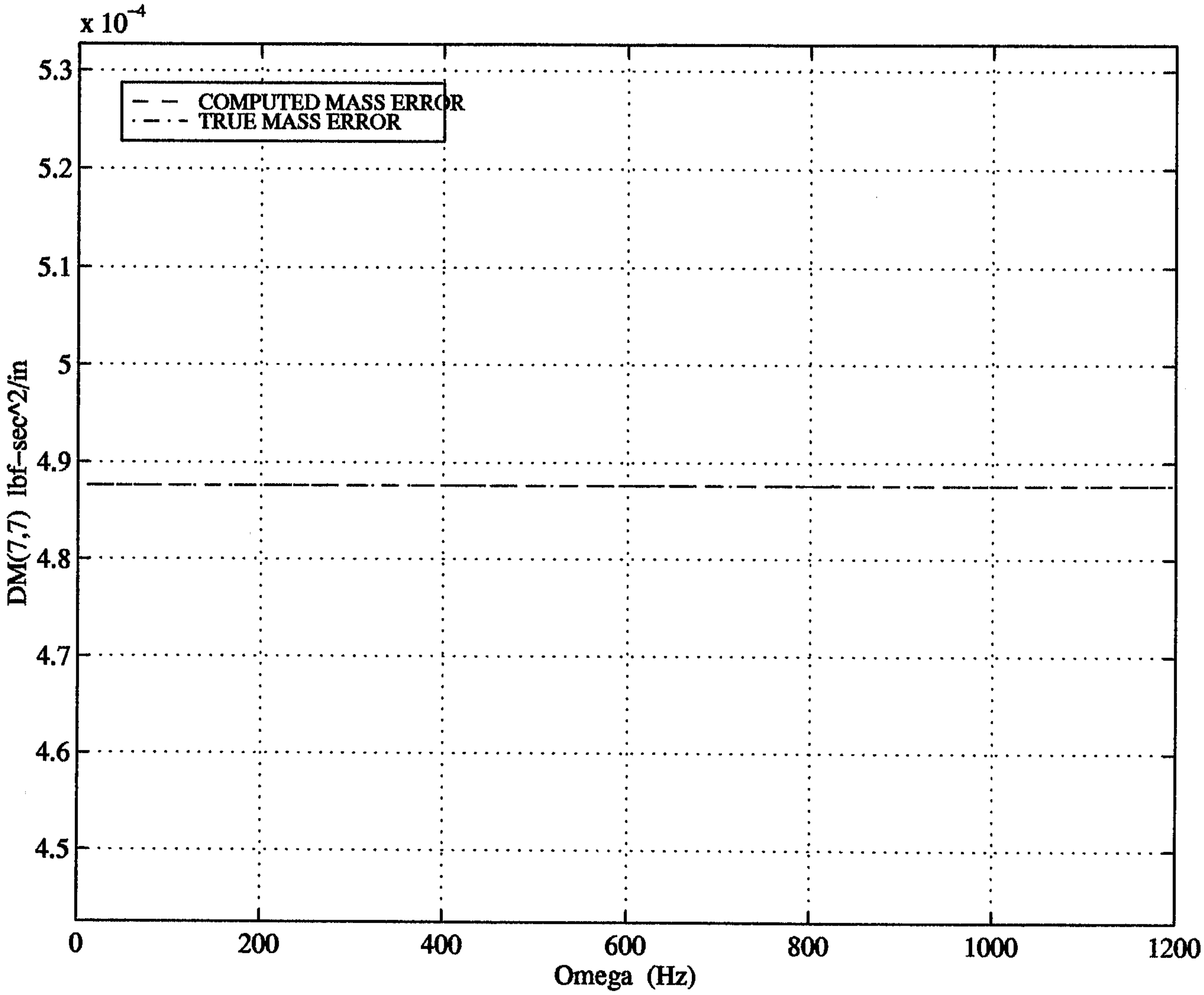

Figure 3- 8 Computed mass vs true mass for spatially complete beam. Plots are identical within plot resolution.

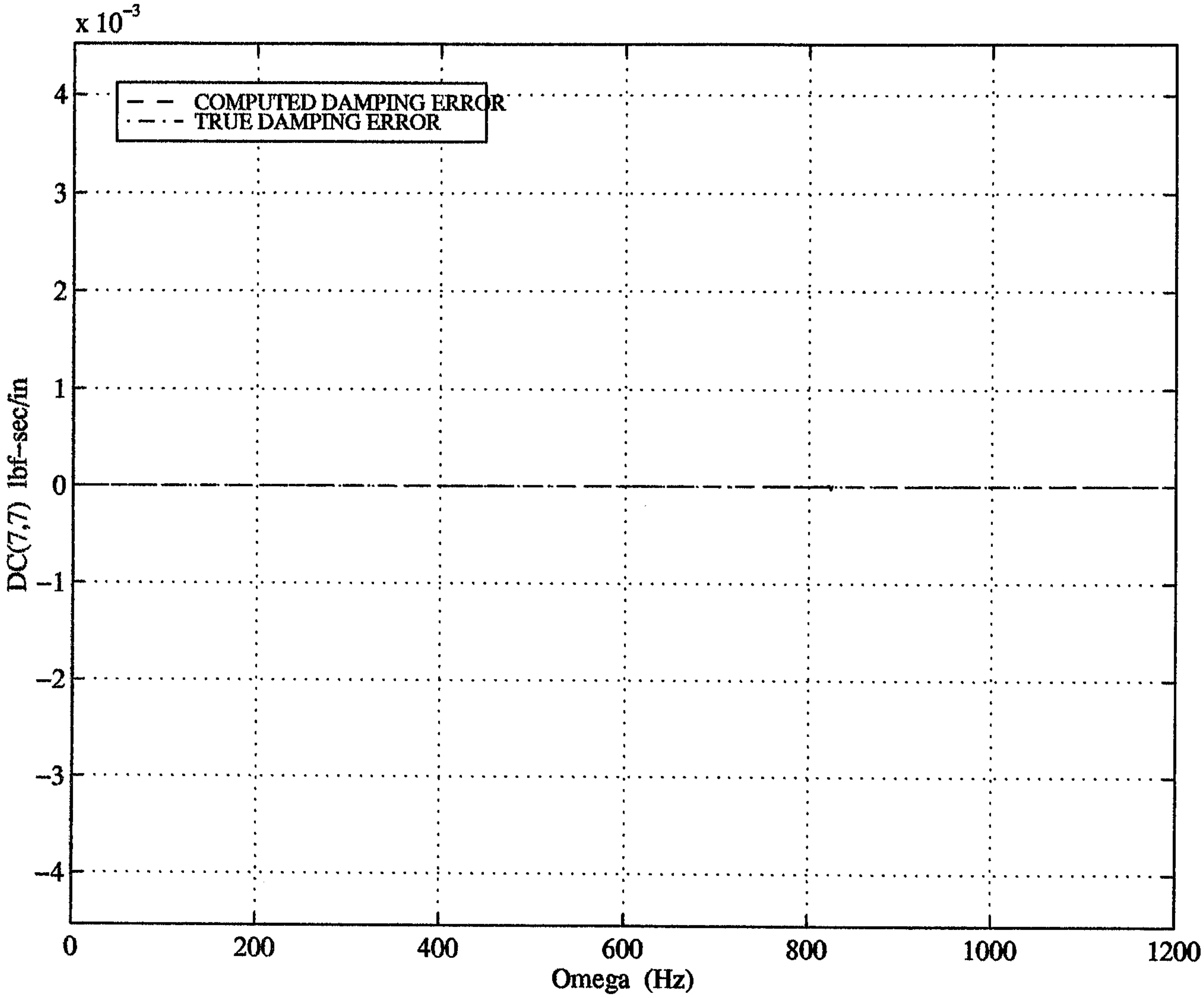

Figure 3- 9 Computed damping vs true damping for spatially complete beam. Plots are identical within plot resolution.

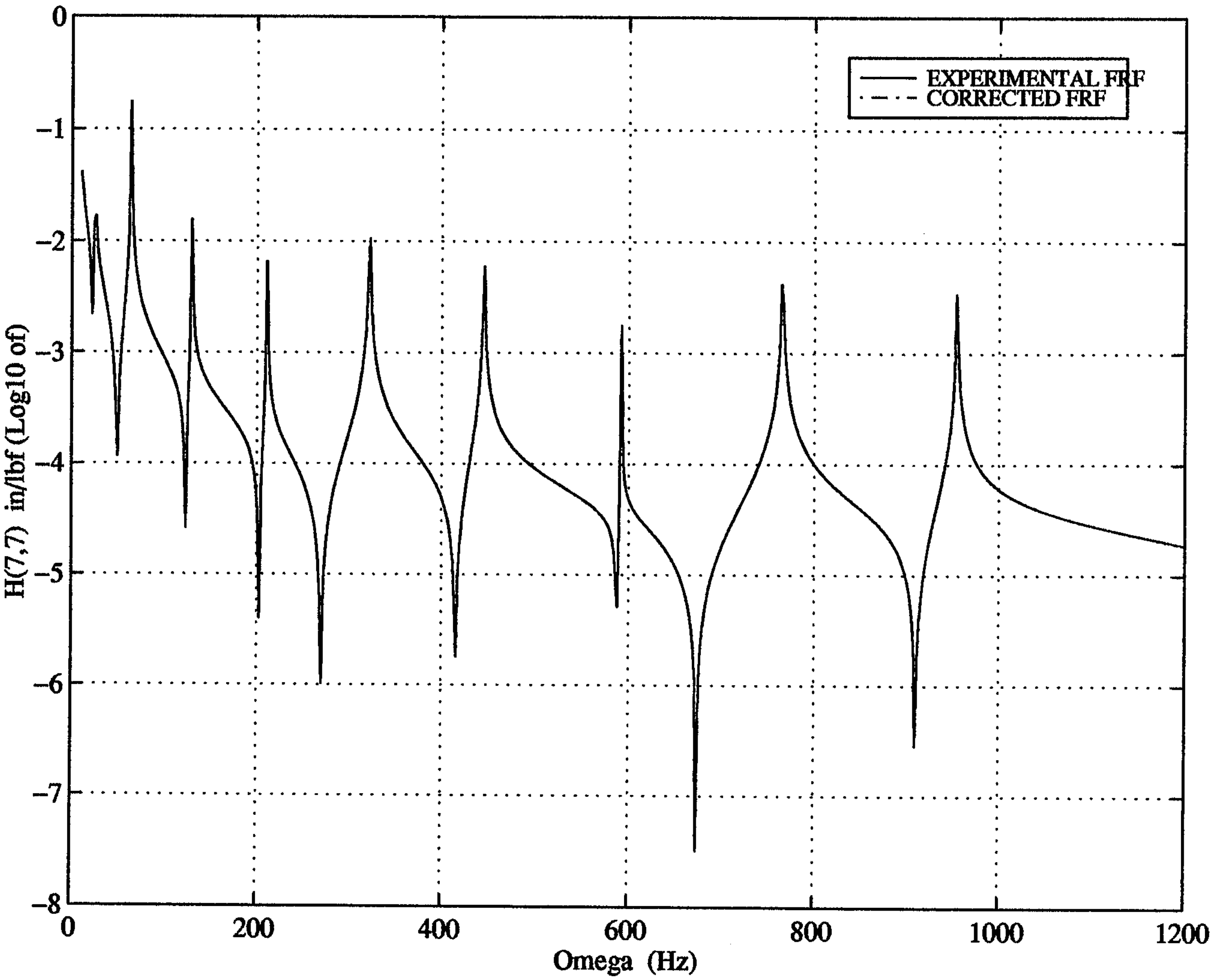

Figure 3- 10 Spatially complete Experimental FRF vs ΔZ corrected FRF. Plots are identical within plot resolution.

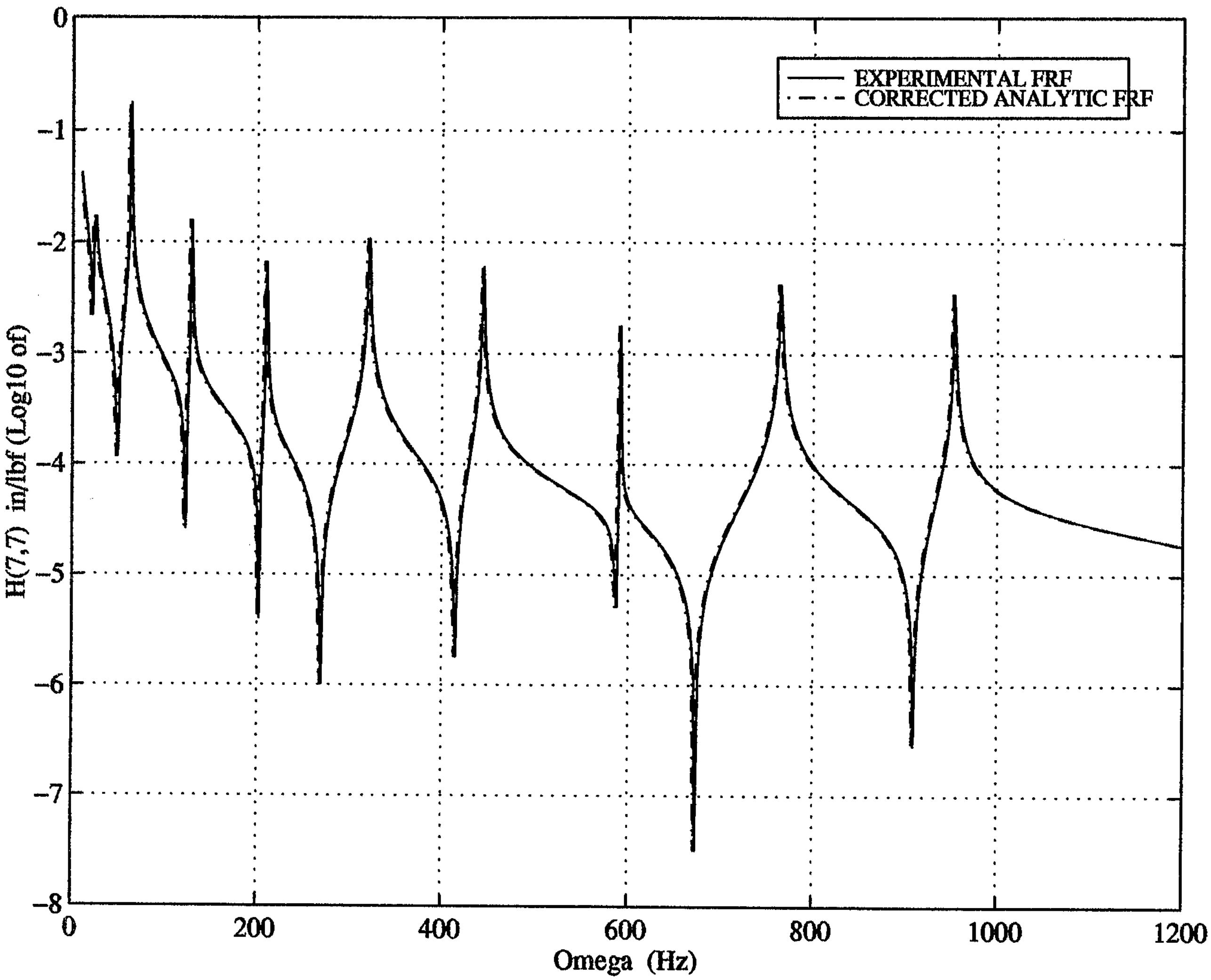

Figure 3- 11 Spatially complete experimental FRF vs ΔK, ΔM, and ΔC corrected FRF.

IV. SPATIALLY INCOMPLETE STRUCTURAL IDENTIFICATION

A. GENERAL DESCRIPTION

To illustrate SST based frequency domain structural identification when the physical system under consideration is spatially incomplete we will again use the FE model of a free-free beam. We simulate the experimental system by imposing a 25% addition to the mass and stiffness of elements 3 and 4 of a 10 element FE model. Figure 4-1 shows the FE modeled beam and the spatially incomplete system that results if FRF data is available only at the displacement DOF of the FE system.

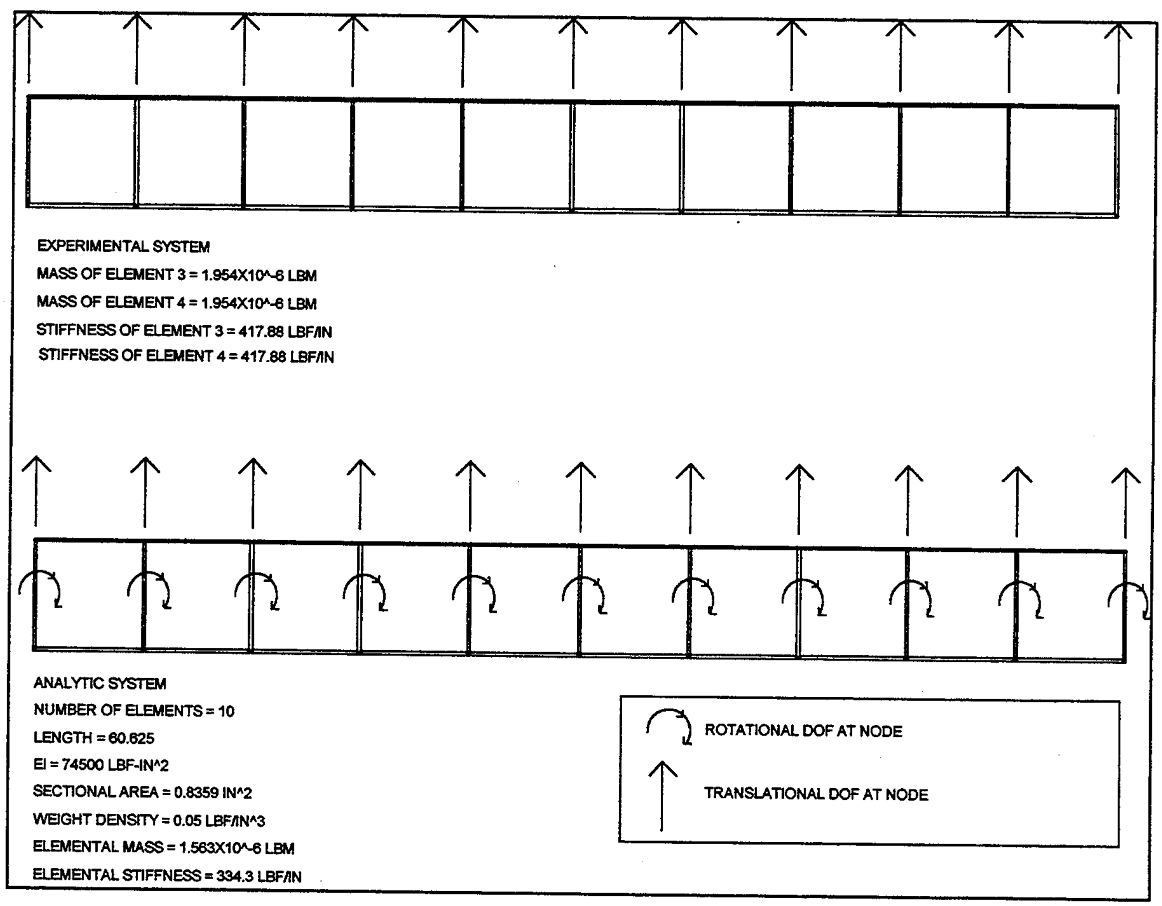

Figure 4- 1 Analytic and experimental spatially incomplete systems.

To obtain a simulated FRF for our spatially incomplete beam we denote by M^a, K^a, and C^a the mass stiffness and damping matrices of the FE model and by M^x, K^x, and C^x the mass stiffness and damping matrices of the experimental system. M^x, K^x, and C^x are obtained by imposing 15% mass and stiffness and a 15% mass additions to the elemental matrices of element 5 and 6 respectively of the FE model. The impedance matrix of the simulated spatially complete beam is given by

$$Z^x(\Omega) = K^x + j\Omega C^x - \Omega^2 M^x \tag{4-1}$$

The FRF matrix of the simulated spatially complete system is given by

$$H^x(\Omega) = Z^x(\Omega)^{-1}. \tag{4-2}$$

We introduce the terminology Analysis set and Omitted set where the Analysis set (A-set) is that set of DOF for which experimental FRF data is available and the Omitted set (O-set) is that set of DOF of the experimental system for which experimental FRF data is unavailable. For our simulated experimental system the A-set consist of the odd numbered translational DOF and the O-set consists of the even numbered rotational DOF, i.e.,

$$A-set = \{1,3,5,\ldots,21\} \tag{4.3a}$$

$$O-set = \{2,4,6,\ldots,22\} \tag{4.3b}$$

We obtain the simulated FRF of our spatially incomplete beam by physically extracting the rows and columns of the simulated spatially complete matrix, H^x, for which FRF data would be available. In our system these are the rotational DOF and all even numbered rows and columns are omitted from the simulated spatially complete FRF to obtain a spatially incomplete FRF which we will denote by $\overline{H^x}$.

For a fixed Ω_0, $\overline{H^x}(\Omega_0)$ is a square matrix of size (length(*A-set*)) by (length(*A-set*)). For our FE model as currently defined, H^a, is of size (number of DOF) by (number of DOF) and is, as is true of most real world cases, of larger size than $\overline{H^x}$. In order to employ the structural synthesis transformation we need to reduce the size of H^a to that of

$\overline{H^x}$. To this end we will consider two reduction methods, FRF matrix extraction [Ref. 1], and the Improved Reduced System as given in [Ref. 2].

B. EXTRACTION REDUCTION METHOD

In order to reduce H^a by the extraction method we simply extract from the full order H those rows and column which correspond to A-set coordinates. Partitioning the impedance and full FRF matrices of our analytical system

$$Z = \begin{bmatrix} Z^a_{aa} & Z^a_{ao} \\ Z^a_{oa} & Z^a_{oo} \end{bmatrix} \tag{4.4a}$$

$$H = \begin{bmatrix} H^a_{aa} & H^a_{ao} \\ H^a_{oa} & H^a_{oo} \end{bmatrix} \tag{4.4b}$$

and using the identity

$$ZH = \begin{bmatrix} Z^a_{aa} & Z^a_{ao} \\ Z^a_{oa} & Z^a_{oo} \end{bmatrix} \begin{bmatrix} H^a_{aa} & H^a_{ao} \\ H^a_{oa} & H^a_{oo} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{4.5}$$

we obtain the relationship

$$H_{aa} = \left(Z_{aa} - Z_{ao} Z_{oo}^{-1} Z_{oa}\right)^{-1} \tag{4.6}$$

We denote the reduced Analytic FRF of Equation (4.6) by $\overline{H^a}$.

C. O-SET SYSTEM

Equation (4.6) relates the extracted reduced order FRF of the analytic model to the impedance of the full order order model. Taking advantage of the identity

$$[Z_{oo}]^{-1} = \left(\det[Z_{oo}]\right)^{-1} adj[Z_{oo}] \tag{4.7}$$

and replacing Z_{oo} by $K_{oo}+j\Omega C_{oo}-j\Omega^2 M_{oo}$ (where det(•) and adj(•) represent the determinant and adjount respectively), we see that an element $\overline{H^a_{ij}}(\Omega)$ is large for those frequencies Ω_o where Ω_o is an eigenvalue of the O-set system, the O-set system being that FE model having stiffness, mass and damping matrices K_{oo}, M_{oo}, and C_{oo} respectively.

D. IMPROVED REDUCTION SYSTEM

To use the improved reduction method (IRS) we first partition Z^a using the A and O sets, then adjust Equation (2.1) to reflect this new coordinate system obtaining

$$\begin{Bmatrix} f_a \\ f_o \end{Bmatrix} = \begin{bmatrix} Z^a_{aa} & Z^a_{ao} \\ Z^a_{oa} & Z^a_{oo} \end{bmatrix} \begin{Bmatrix} x_a \\ x_o \end{Bmatrix} \tag{4.8}$$

Expanding Equation (4.8) into two equations yields

$$f_a = Z^a_{aa} x_a + Z^a_{ao} x_o \tag{4.9a}$$

$$f_o = Z^a_{oa} x_a + Z^a_{oo} x_o \tag{4.9b}$$

O-set coordinates are not associated with FRF data measurement locations on the physical structure, therefore the forcing function at O-set coordinates can be set to zero. Making these substitution into Equation (4.9b) and solving for the generalized structural response coordinates leads to:

$$x_o = -Z^{-1}_{oo} Z^a_{oa} x_a \tag{4.10a}$$

$$\begin{Bmatrix} x_a \\ x_o \end{Bmatrix} = \begin{bmatrix} I \\ -Z^{-1}_{oo} Z^a_{oa} \end{bmatrix} \{x_a\} \tag{4.10b}$$

Substituting these results into Equation (4.8) yields,

$$\begin{Bmatrix} f_a \\ 0 \end{Bmatrix} = \begin{bmatrix} Z^{aa}_{aa} & Z^o_{aa} \\ Z^a_{oa} & Z^a_{oo} \end{bmatrix} \begin{bmatrix} I \\ -Z^{a\ -1}_{oo} Z^a_{oa} \end{bmatrix} \{x_a\} \tag{4.11}$$

hence

$$\{f_a\} = \left[Z_{aa} - Z_{ao} Z^{-1}_{oo} Z_{oa} \right] \{x_a\} \tag{4.12}$$

When Ω=0 the Equation (4.10a) yields the static reduction relationship between omitted and retained coordinates and is given by

$$\{x_o\} = \left[-K^{-1}_{oo} K_{oa} \right] \{x_a\} \tag{4.13}$$

The IRS relationship is given by

$$\{x_o\} = \left[-K_{oo}^{-1}K_{oa} + TM_{stat}^{-1}K_{stat}\right]\{x_a\} \tag{4.14}$$

where

$$T = K_{oo}^{-1}M_{oa} - K_{oo}^{-1}M_{oo}K_{oo}^{-1}K_{oa} \tag{4.15}$$

and K_{stat} and M_{stat} are the statically reduced [Ref. 2, 3] stiffness and mass matrices.

Unlike the spatially complete case where we only had to consider two system (the analytic and the experimental systems), in the case of a spatially incomplete system there are actually five systems with which we must concern ourselves [Ref 4]: the analytic system, the experimental system, the reduced analytic system that results from conducting dynamic reduction on the mass and stiffness matrices of the analytic system, and the omitted systems of both the analytic and the experimental system. Table 4-1 shows the frequencies of each the first four of these systems.

Figure 4-2 shows a comparision of the analytic and experimental FRF of our spatially incomplete beam. We see that by using IRS reduction of the analytic system those modes above approximately 1000 Hz i.e., those modes associated with the reduced out rotations, are not present. Figure 4-3 shows a similar comparision where we have used extraction reduction and the higher modes are present. In all that follows we shall use IRS reduction of our analytical systems. Figure 4-4 shows the localization matrix diagonal at Ω=196.1 Hz while Figure 4-5 shows the frequency dependency of the localization diagonals over the frequency range our our spatially incomplete beam. For a spatially incomplete system the determination of the locations of the error coordinates is not a clearly defined task. We shall not discuss the problem of actually determining the exact error set in the spatially incomplete case and will use our knowledge of the true location of the error coordinates to aid in our localization. As we shall make use of the concept of the size of the error set again, we will simply note that using a reduction method like IRS causes errors to be 'smeared' in the reduced analytic model. Table 4-2 shows the A-set and O-set of our simulated beam along with the locations of the true errors as well as the

computed C-set at Ω=196 Hz. The computed C-set is the set of all DOF having a diagonal entry whose absolute value exceeds a given tolerance.

Mode	Anal (Hz)	Exp (Hz)	Reduced (Hz)	O set (Hz)
1	25.41	25.83	25.41	1244.28
2	70.07	70.85	70.07	1287.94
3	137.45	137.93	137.45	1417.14
4	227.55	227.37	227.55	1629.59
5	340.84	341.14	340.84	1928.6
6	478.0	477.97	478.0	2327.83
7	639.9	639.92	640.3	2853.66
8	826.5	826.24	829.7	3541
9	1026.8	1025.99	1029.74	4399.32
10	1363.62	1364.51		5280.86
11	1640.62	1640.71		5701.99
12	1981.19	1984.61		
13	2381.4	2380.32		
14	2850.01	2847.38		
15	3397.39	3397.74		
26	4027.82	4026.28		
17	4720.15	4738.64		
18	5376.83	5380.09		
19	6794.91	6787.09		
20	6807.52	6804.12		

Table 4-1 Analytic, Experimental, Reduced, and Omitted System Frequencies of Spatially Incomplete Beam.

A Set	1	3	5	7	9	11	13	15	17	19	21
O Set	2	4	6	8	10	12	14	16	18	20	22
True C Set	5	6	7	8	9	10					
Computed C Set	1	3	5	7	9	11	13				

Table 4-2 Analytic set, Omitted set, Computed C set, and True C set DOF for spatially incomplete beam.

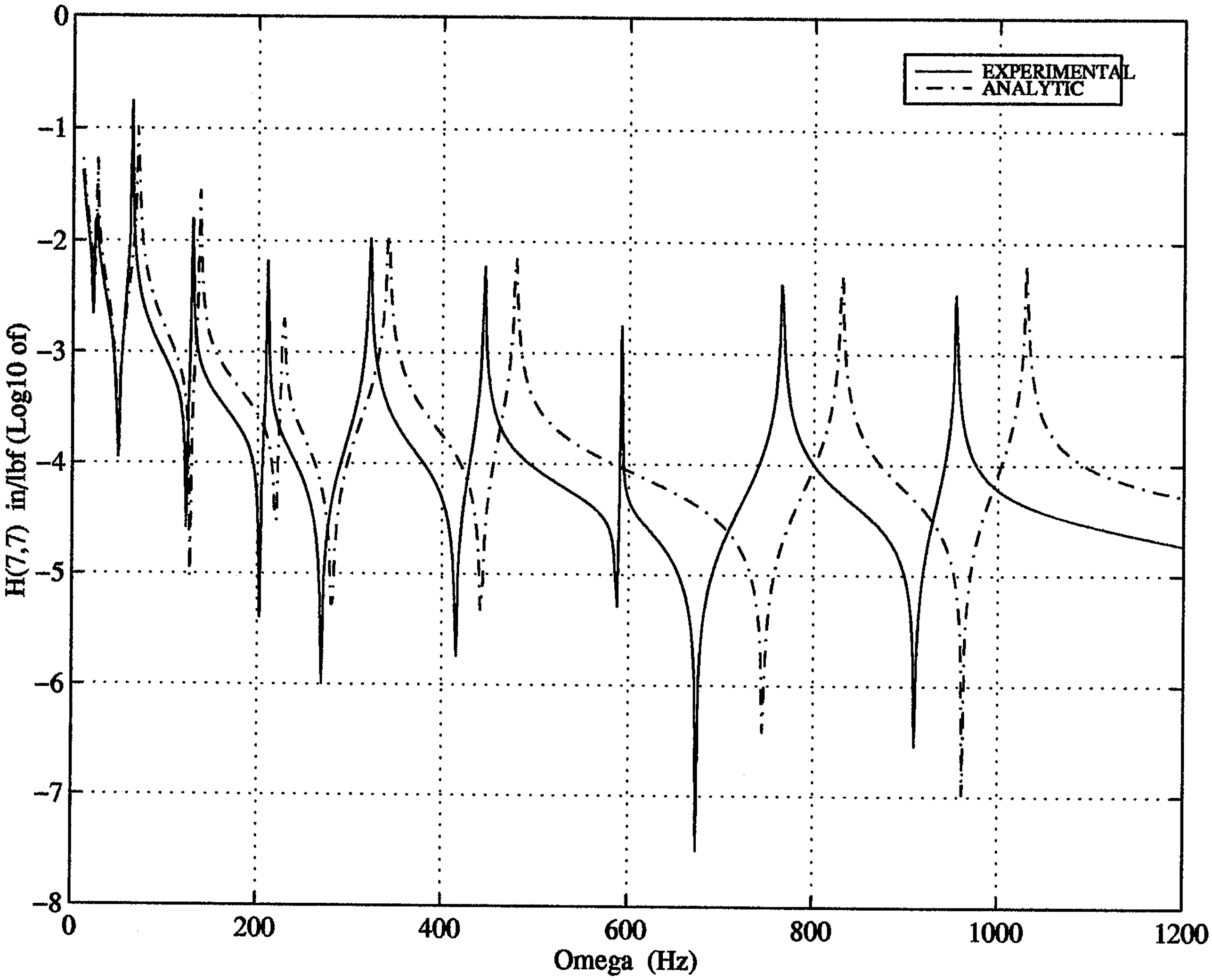

Figure 4- 2 Analytic FRF vs experimental FRF for spatially incomplete beam using IRS reduction.

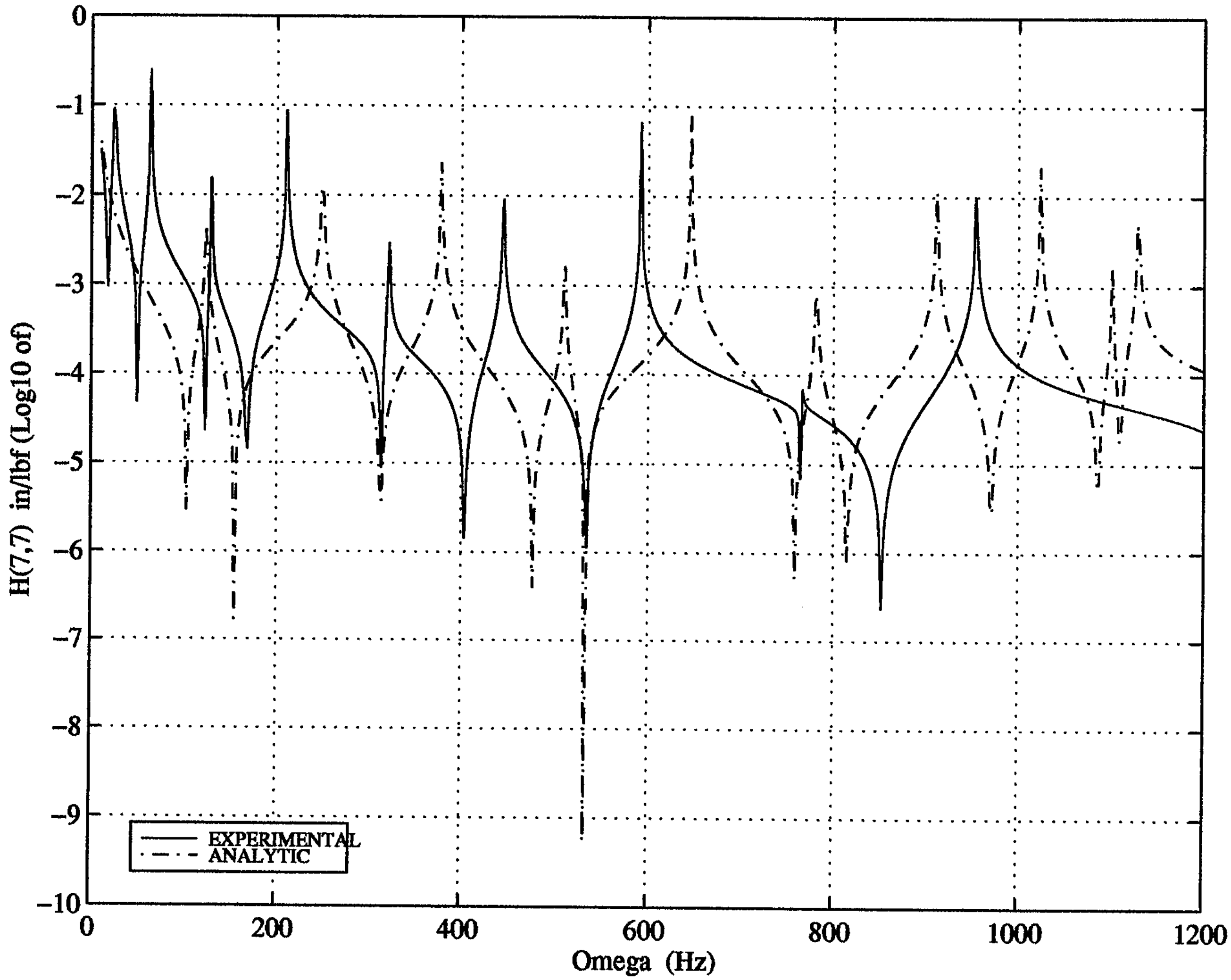

Figure 4- 3 Analytic FRF vs experimental FRF for spatially incomplete beam using extraction reduction.

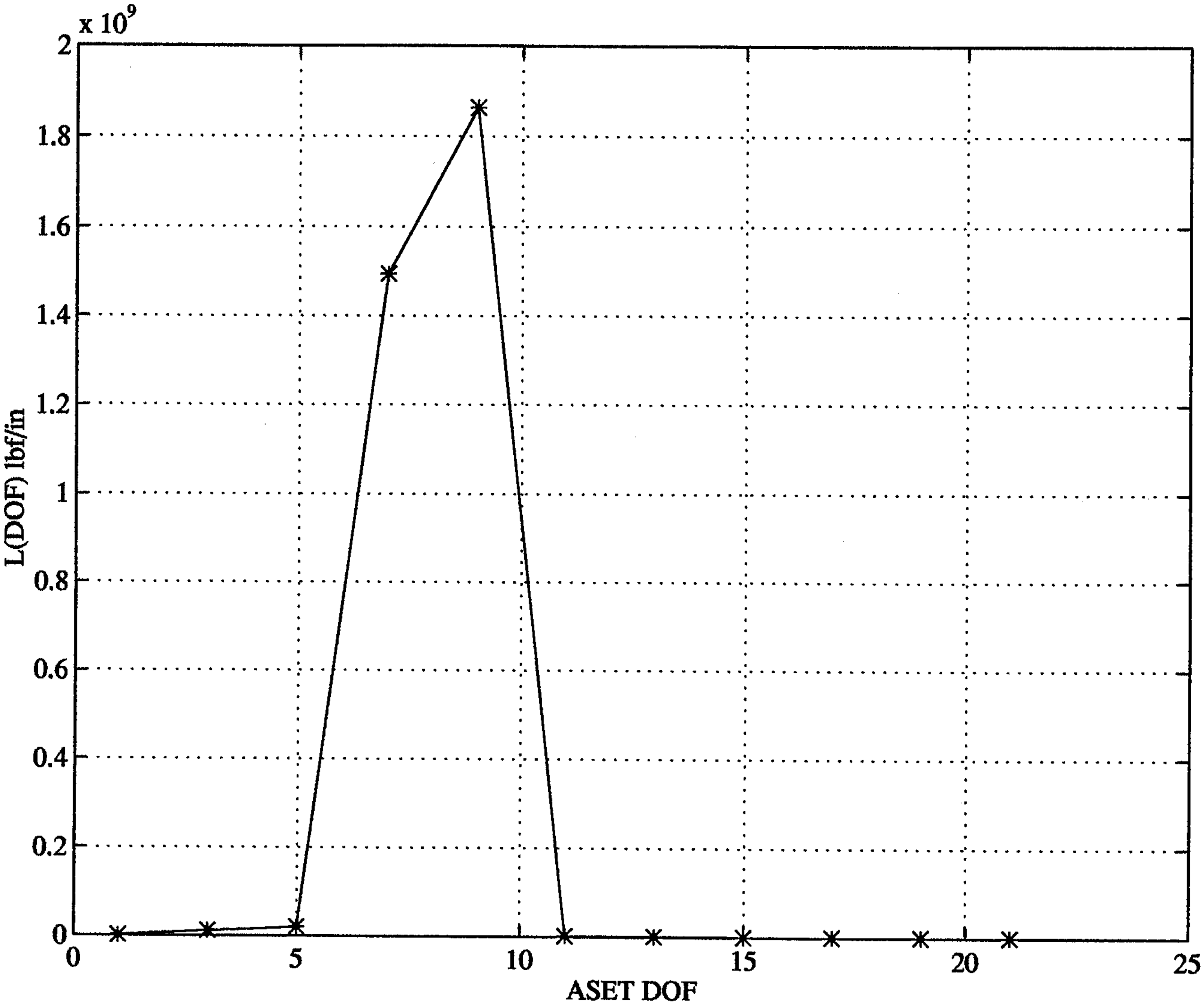

Figure 4- 4 Localization matrix diagonal at Ω=196.1 Hz for spatially incomplete beam.

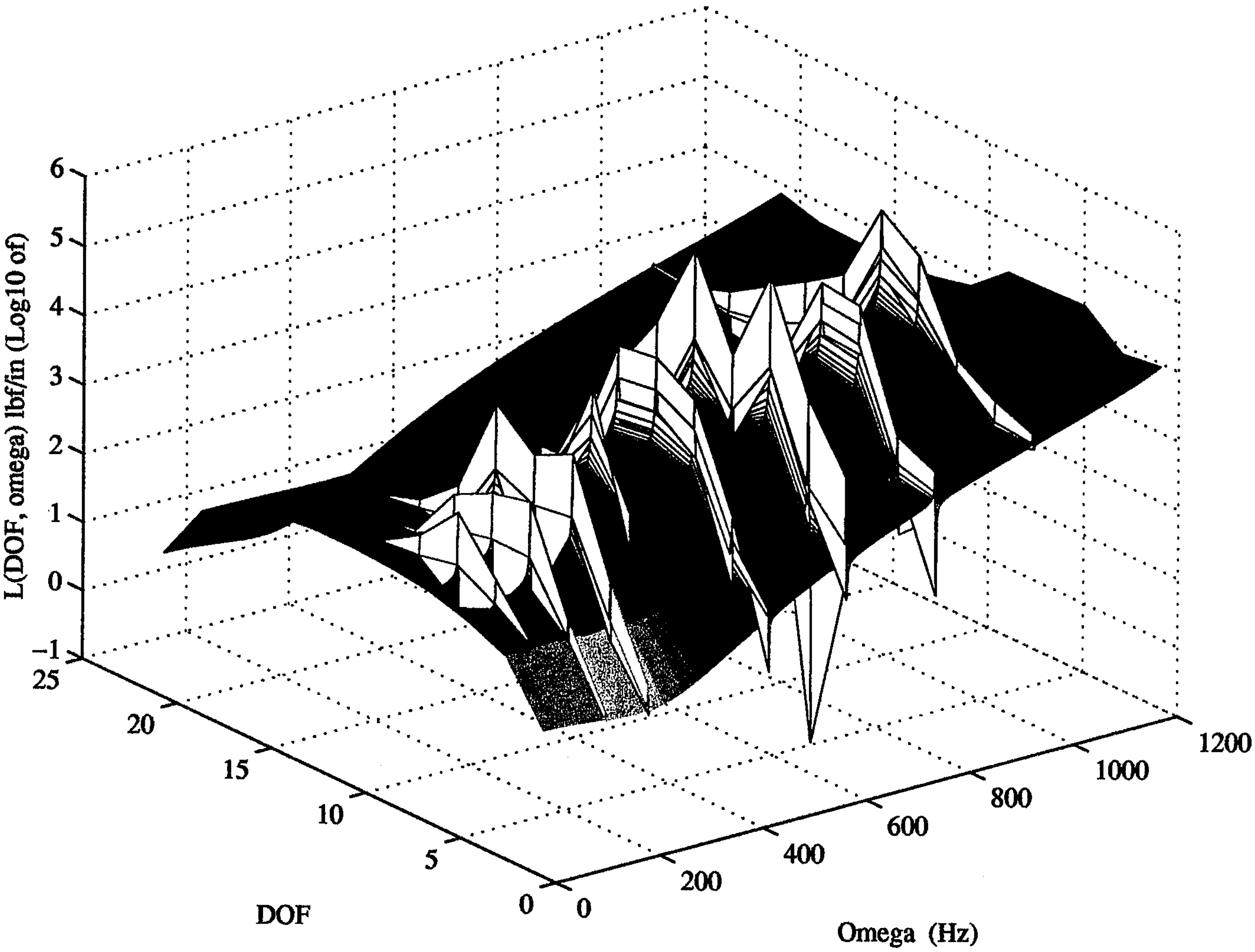

Figure 4- 5 Frequency dependence of localization matrix diagonals for spatially incomplete beam.

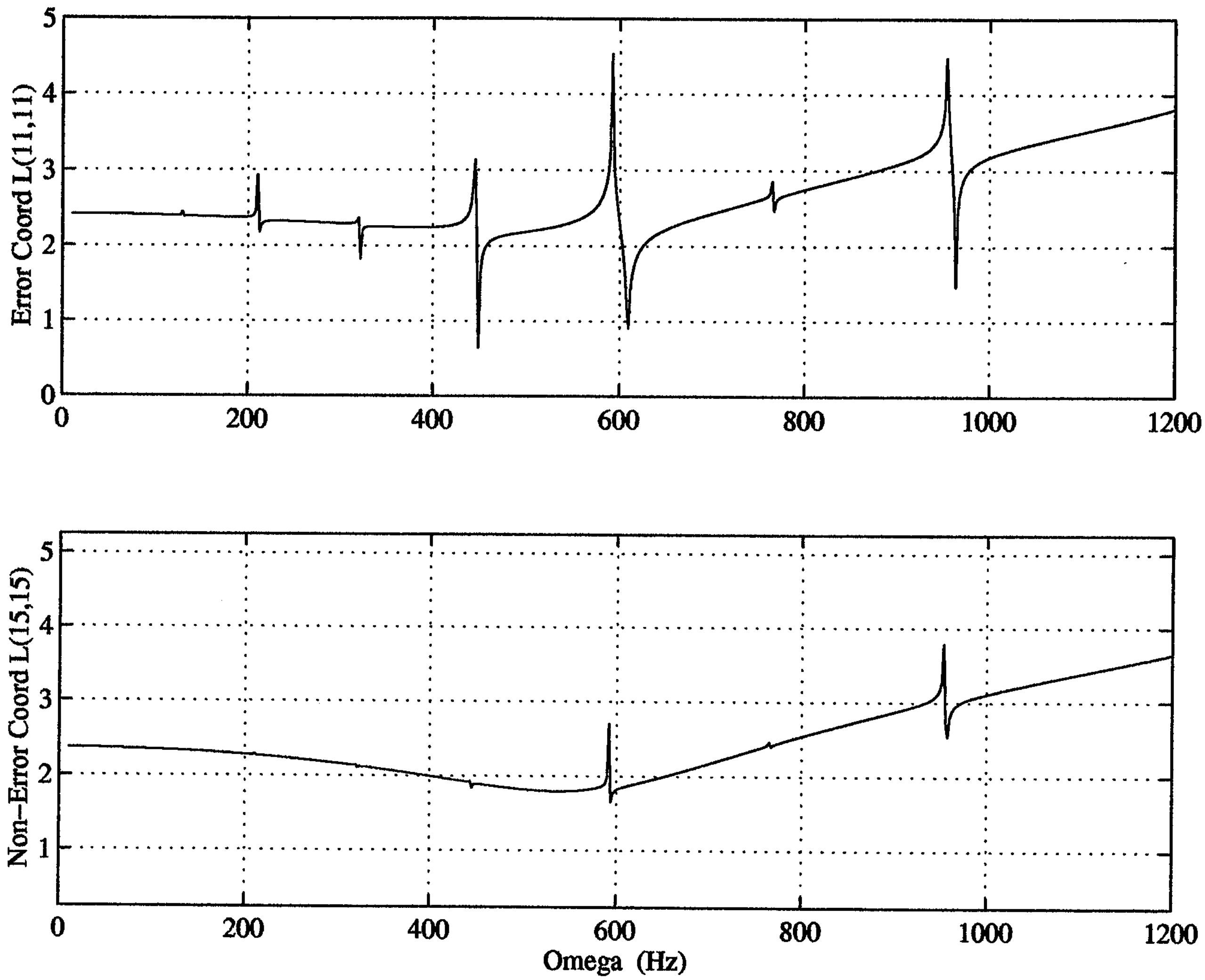

Figure 4- 6 Frequency dependence of error and non error set localization matrix diagonals for spatially incomplete beam. Units are lbf/in (Log10 of).

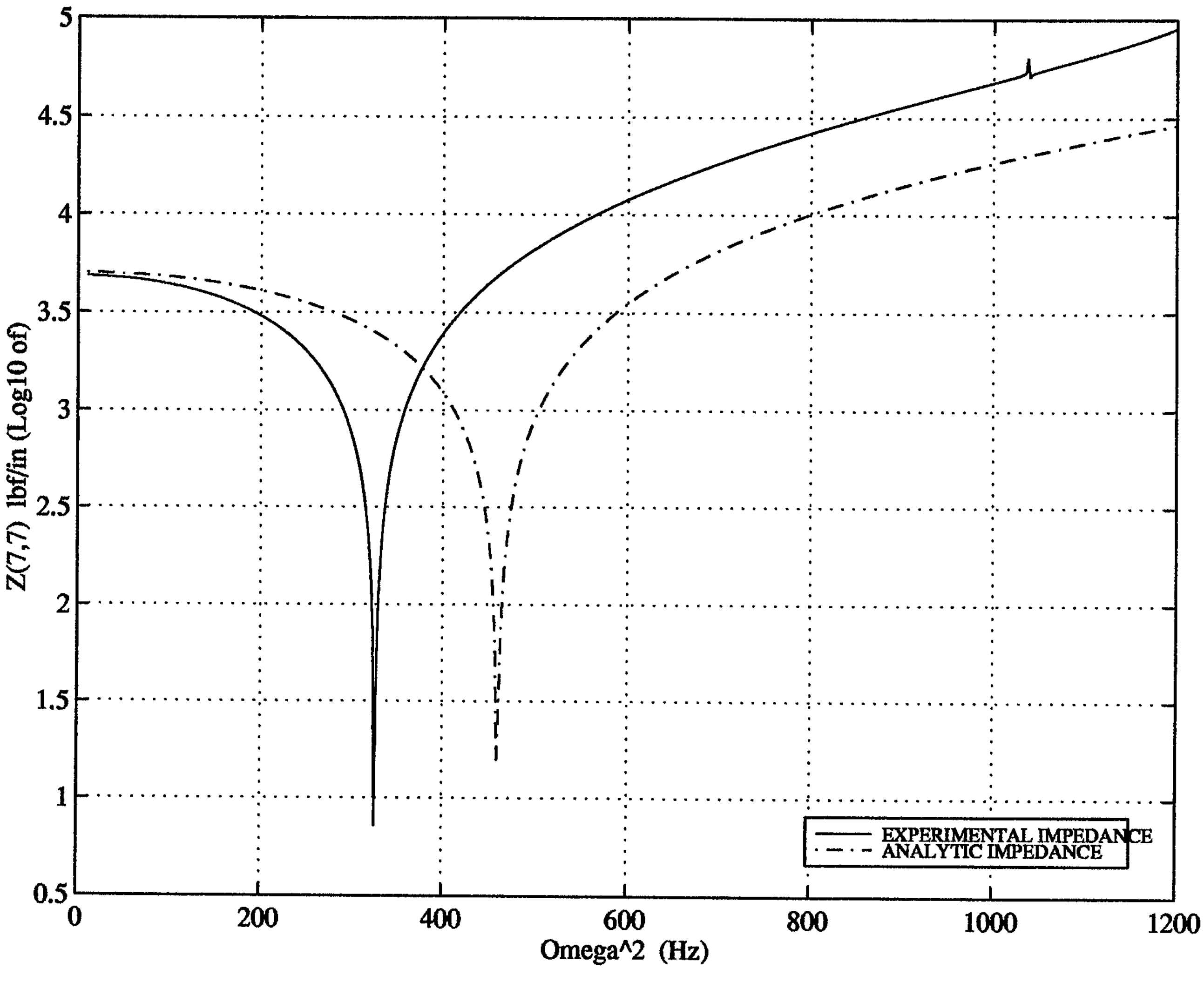

Figure 4- 7 Analytic vs experimental impedance for spatially incomplete beam.

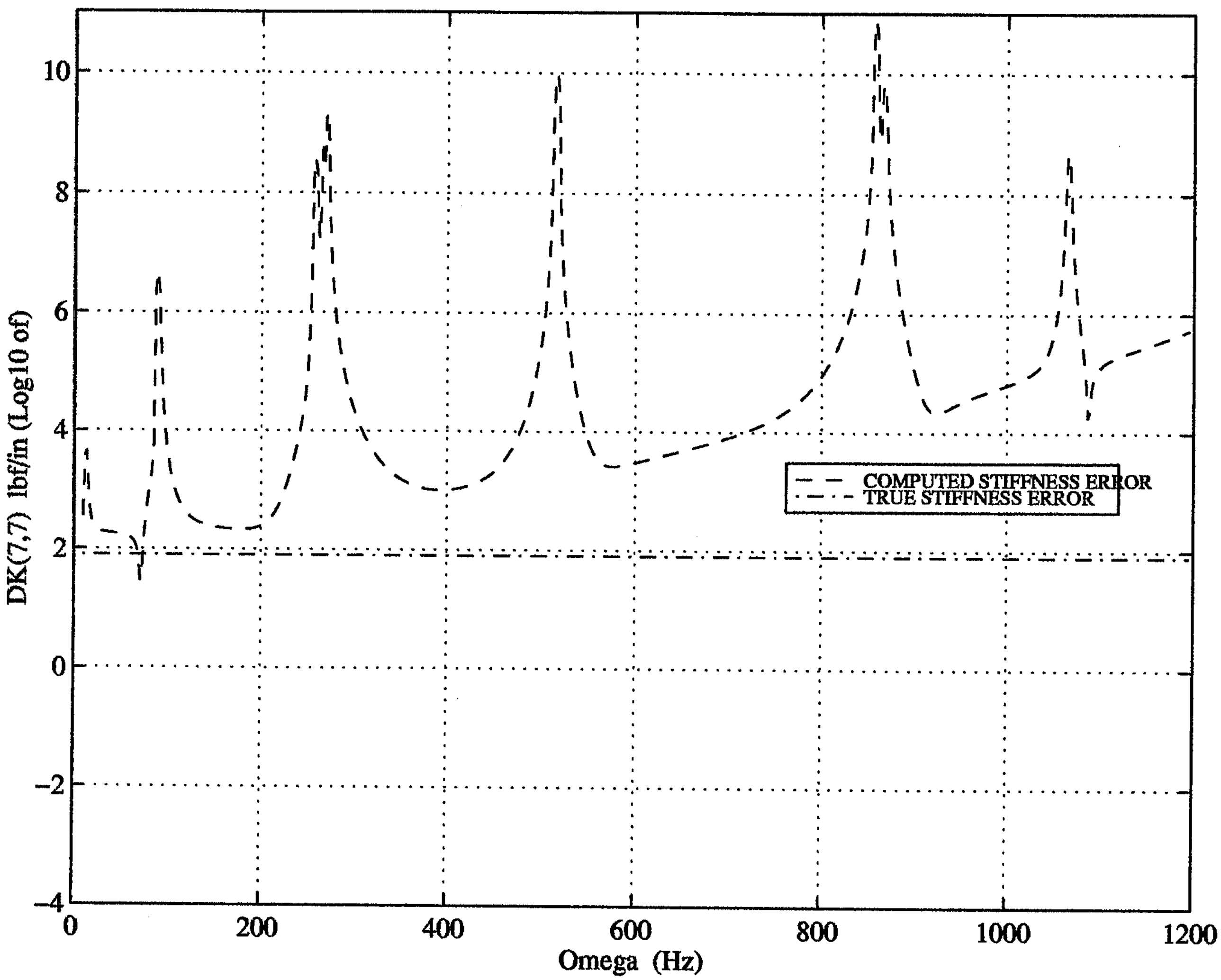

Figure 4- 8 Computed Stiffness vs True Stiffness for spatially incomplete beam.

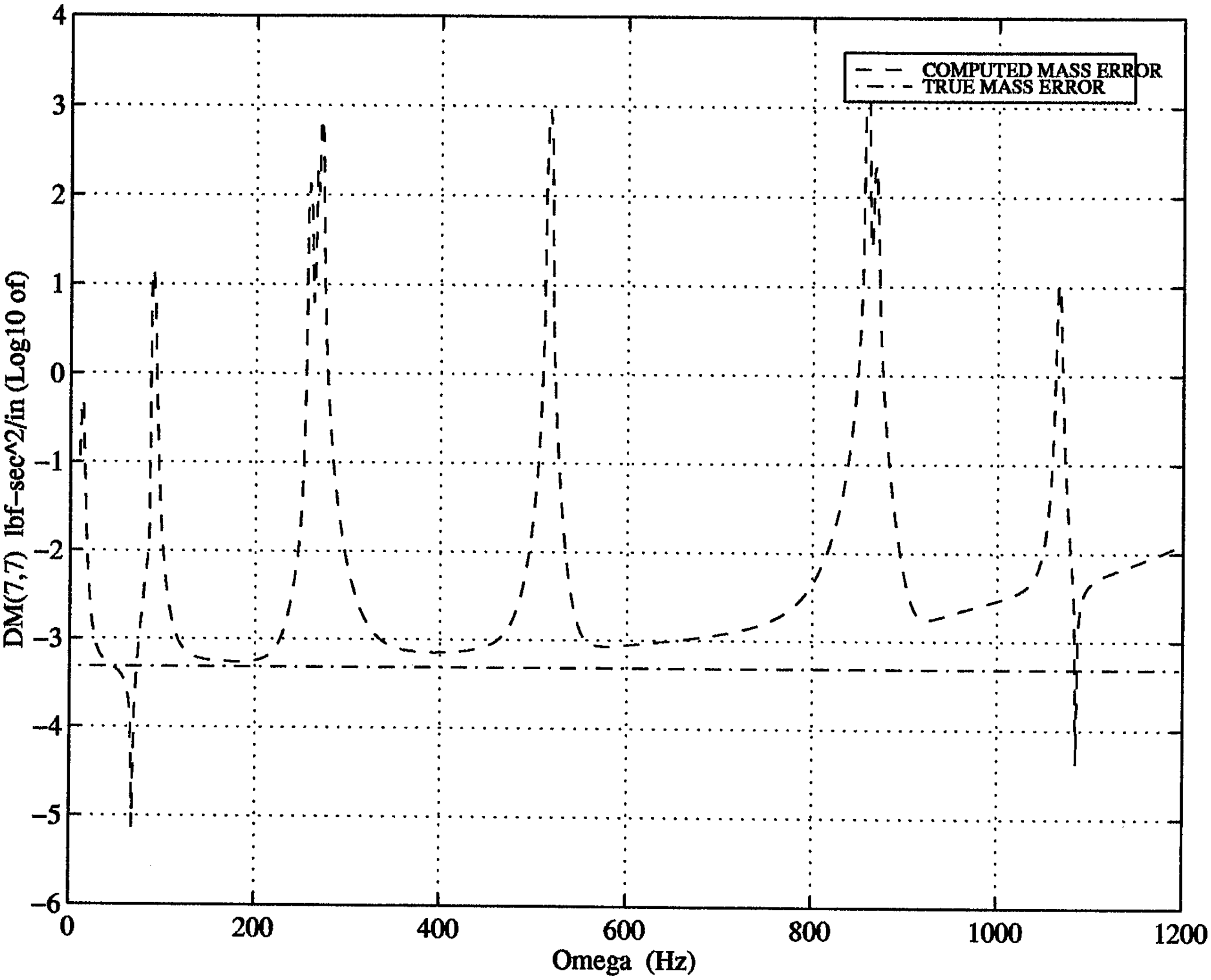

Figure 4- 9 Computed Mass vs True Mass for spatially incomplete beam.

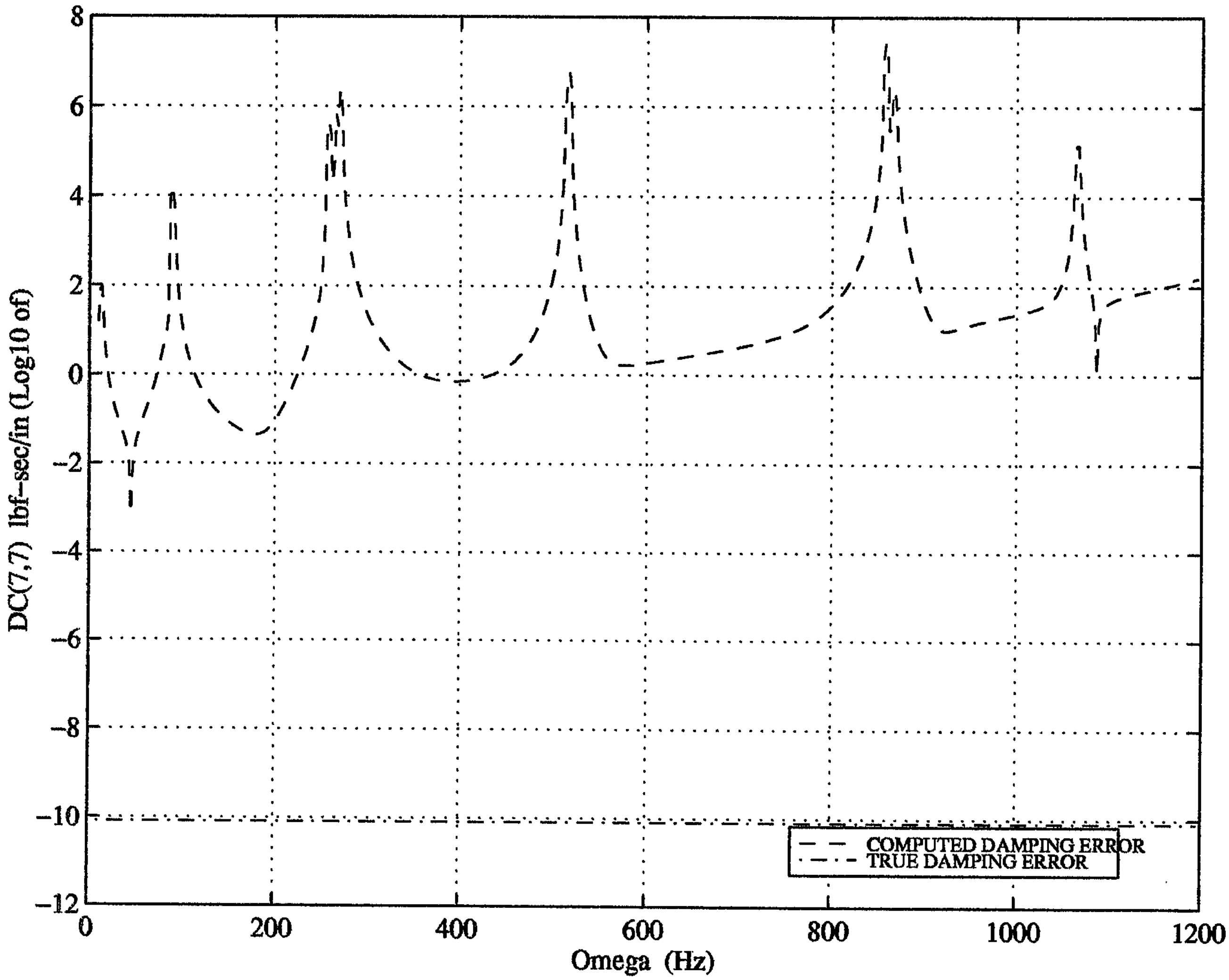

Figure 4- 10 Computed Damping vs True Damping for spatially incomplete beam.

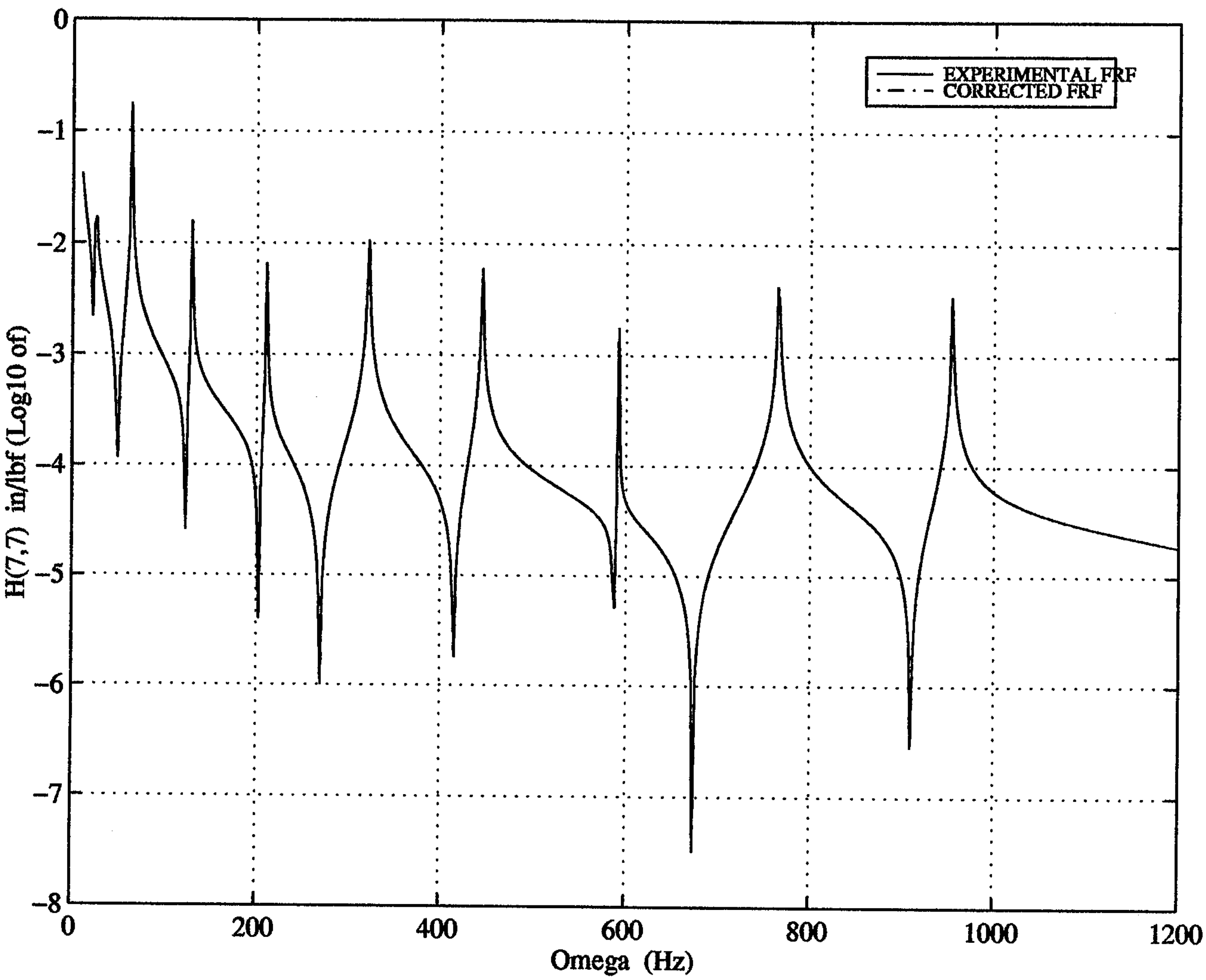

Figure 4- 11 ΔZ Corrected FRF vs experimental FRF for spatially incomplete beam. Plots are identical to within plot resolution.

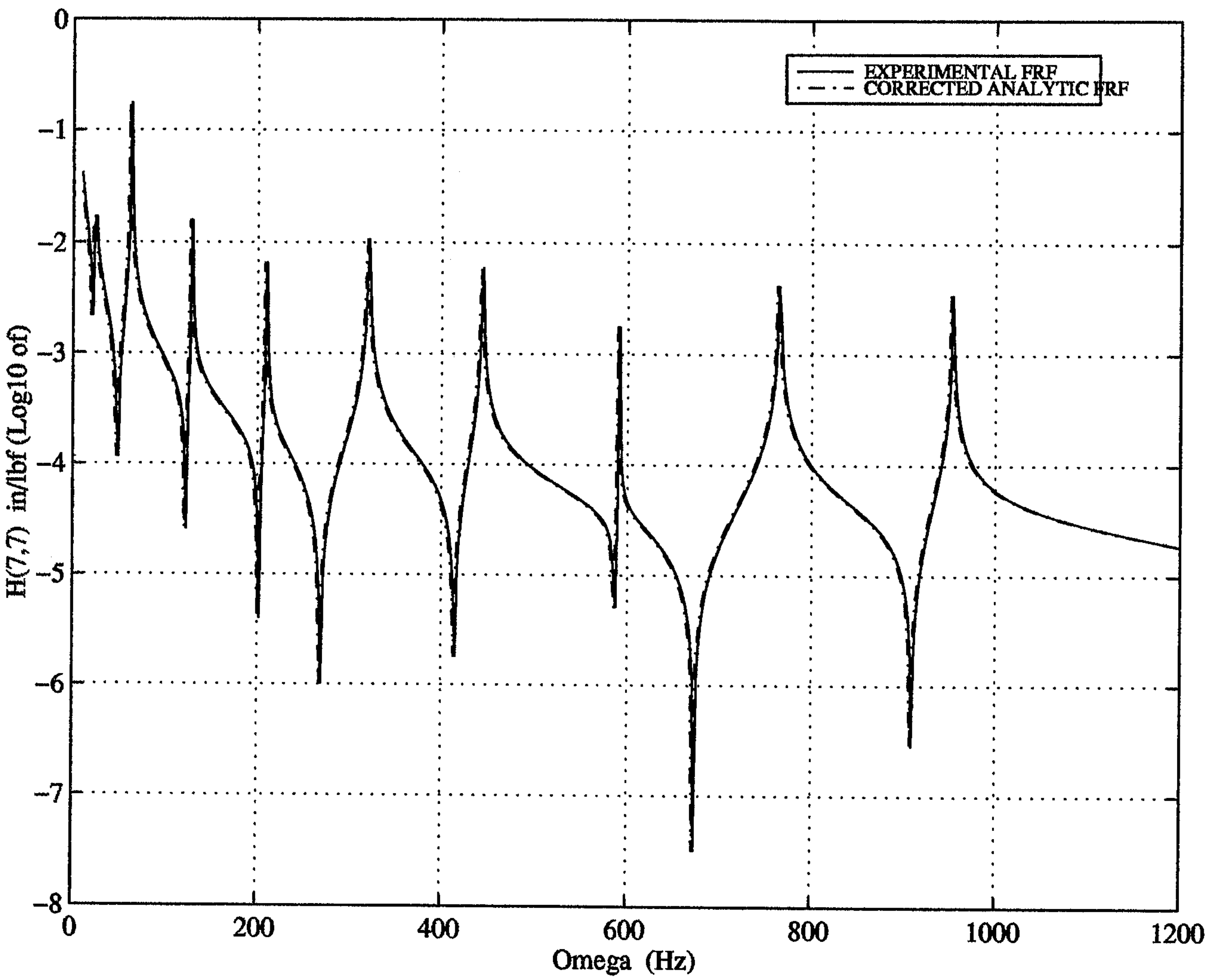

Figure 4- 12 ΔK/ΔM/ΔC Corrected FRF vs experimental FRF for spatially incomplete beam. The offset of the two plots is equal to the sampling frequency ΔΩ.

V. SINGLE MODE SOLUTIONS

A. SINGLE MODE MATRIX SOLUTIONS

In the case of a spatially complete system we have seen that the SST yielded frequency independent error matrices ΔK, ΔM, and ΔC that could be used to correct the stiffness, mass, and damping matices of the FE model in such a manner that the FRF of the corrected FE model was exactly equal to that of the experimental system. In the case of a spatially incomplete system the constituent solutions ΔK, ΔM, and ΔC given in chapters III and IV were in general frequency dependent solutions which serve only as corrections to the reduced FE model. Ideally we seek frequency independent solutions which are corrections to the full FE model. For now we shall only deal with the simpler problem of trying to find frequency independent solutions which serve as corrections to the reduced FE model.

We shall employ Equation (2.30). For a given experimental system mode, ω_n, consider a frequency bandwidth $[\Omega_l, \Omega_u]$ such that $\omega_n \in [\Omega_l, \Omega_u]$. Let $\Xi=\{\Omega_1,\Omega_2,\ldots,\Omega_m\}$ be a frequency sampling of the bandwidth $[\Omega_l, \Omega_u]$, i.e., $\Omega_l \le \Omega_i \le \Omega_u$ for each $\Omega_i \in \Xi$. For each $\Omega_i \in \Xi$ we can form the error impedance matrix $\Delta Z(\Omega_i)$. We apply Equation (2.30) to the partition yeilding the following system of m equations in three unknowns:

$$\begin{bmatrix} \Delta Z_c(\Omega_1) \\ \vdots \\ \Delta Z_c(\Omega_i) \\ \vdots \\ \Delta Z_c(\Omega_m) \end{bmatrix} = \begin{bmatrix} I & -\Omega_1^2 I & j\Omega_1 I \\ \vdots & \vdots & \vdots \\ I & -\Omega_i^2 I & j\Omega_i I \\ \vdots & \vdots & \vdots \\ I & -\Omega_m^2 I & j\Omega_m I \end{bmatrix} \begin{bmatrix} \Delta K_c^{\omega_n} \\ \Delta M_c^{\omega_n} \\ \Delta C_c^{\omega_n} \end{bmatrix} \tag{5.1}$$

We will demonstrate by example that for properly chosen bandwidths $[\Omega_l, \Omega_u]$, the solution $\Delta K_c^{\omega_n}$, $\Delta M_c^{\omega_n}$, $\Delta C_c^{\omega_n}$ of Equation (5.1) approximately corrects the Analytic FRF over the frequency bandwidth $[\Omega_l,\Omega_u]$, i.e., if $H^x(\Omega)$ is the value of the FRF matrix of the experimental system at a frequency Ω, $\overline{H^a}(\Omega)$ the value of the FE reduced analytic FRF

matrix at Ω, and $\overline{H^a_{corr}}(\Omega)$ the value of the single mode corrected reduced Analytic FRF matrix at Ω for an experimental system mode ω_n where $\Omega \in [\Omega_l, \Omega_u]$ then

$$\left|H^x(\Omega) - \overline{H^a_{corr}}(\Omega)\right| \leq \left|H^x(\Omega) - \overline{H^a}(\Omega)\right| \tag{5.2}$$

We will refer to the solution $\Delta K_c^{\omega_n}$, $\Delta M_c^{\omega_n}$, $\Delta C_c^{\omega_n}$ as a single mode solution at ω_n. In the case of a spatially complete system, such as the spatially complete beam discussed in chapter III, all single mode solutions are found to be identical to the unique frequency independent solution that was obtained in chapter III, hence the corrections are valid throughout the frequency range of the experimental system and the left hand side of Equation 5.2 is zero. For a spatially incomplete system this is not the case.

In support of Equation (5.2) we chose a 1 Hz frequency bandwidth centered on mode 1 (ω_1=25.21 Hz) of our spatially incomplete beam as defined in Chapter III. For our frequency sampling, Ξ, we will use a sampling frequency of .5 Hz which results in 3 points which are equally spaced over the interval, Ξ={24.7178 Hz, 25.2178 Hz, 25.7178 Hz}. Solving Equation (5.1) we get

$$\Delta K_c^{\omega_1} = \begin{bmatrix} -3.91 & -13.60 & 21.70 \\ -13.60 & 170.5 & -187.5 \\ -21.70 & -187.5 & 194.6 \end{bmatrix} \tag{5.3a}$$

$$\Delta M_c^{\omega_1} = \begin{bmatrix} -0.0001 & 0.0006 & -0.0005 \\ 0.0006 & -0.0033 & 0.0026 \\ -0.0005 & -0.0026 & -0.0016 \end{bmatrix} \tag{5.3b}$$

$$\Delta C_c^{\omega_1} = \begin{bmatrix} 0+0.0270j & 0-0.2291j & 0+0.2277j \\ 0-0.2291j & 0+1.5078j & 0-1.4627j \\ 0+0.2277j & 0-1.4627j & 0+1.3781j \end{bmatrix} \tag{5.3c}$$

Figure 5-1 shows a comparison of experimental, uncorrected and mode 1 corrected FRFs for our spatially incomplete beam using a 1 Hz frequency bandwidth centered on Mode 1 and a frequency sampling consisting of 3 frequencies equally spaced over the bandwidth. Figure 5-2 shows the results of including mode 1 and mode 2 in the

frequency bandwidth [Ω_l, Ω_u]. As Figure 5-2 shows Equation 5.1 is not as accurate when multiple frequencies are included in the bandwidth. Figure 5-3 is a comparison plot of the experimental and analytic FRF versus the single mode matrix solution corrected FRFs over 25, 10, and 1 Hz bandwidths using a 3 point frequency sampling. Figure 5-3 shows that the precedure is sensitive to the size of the bandwidth over which the solution is computed, i.e., better accuracy is achieved with smaller bandwidths. Figure 5-4 is a comparison plot of the experimental and uncorrected analytic FRFs versus the single mode matrix solution corrected FRFs computed over a 1 Hz bandwidth using frequency samplings of 200, 50, 10, and 3 points equally spaced over the bandwidth. Figure 5-4 shows that the procedure is fairly insensitive to sampling frequency.

Figure 5-5 is a comparison plot of the experimental and analytic FRFs versus the single mode solution corrected FRF for a 1 Hz bandwidth using a 3 point frequency sampling in the case of a spatially complete beam. As Figure 5-5 shows the procedure is exact for spatially complete systems.

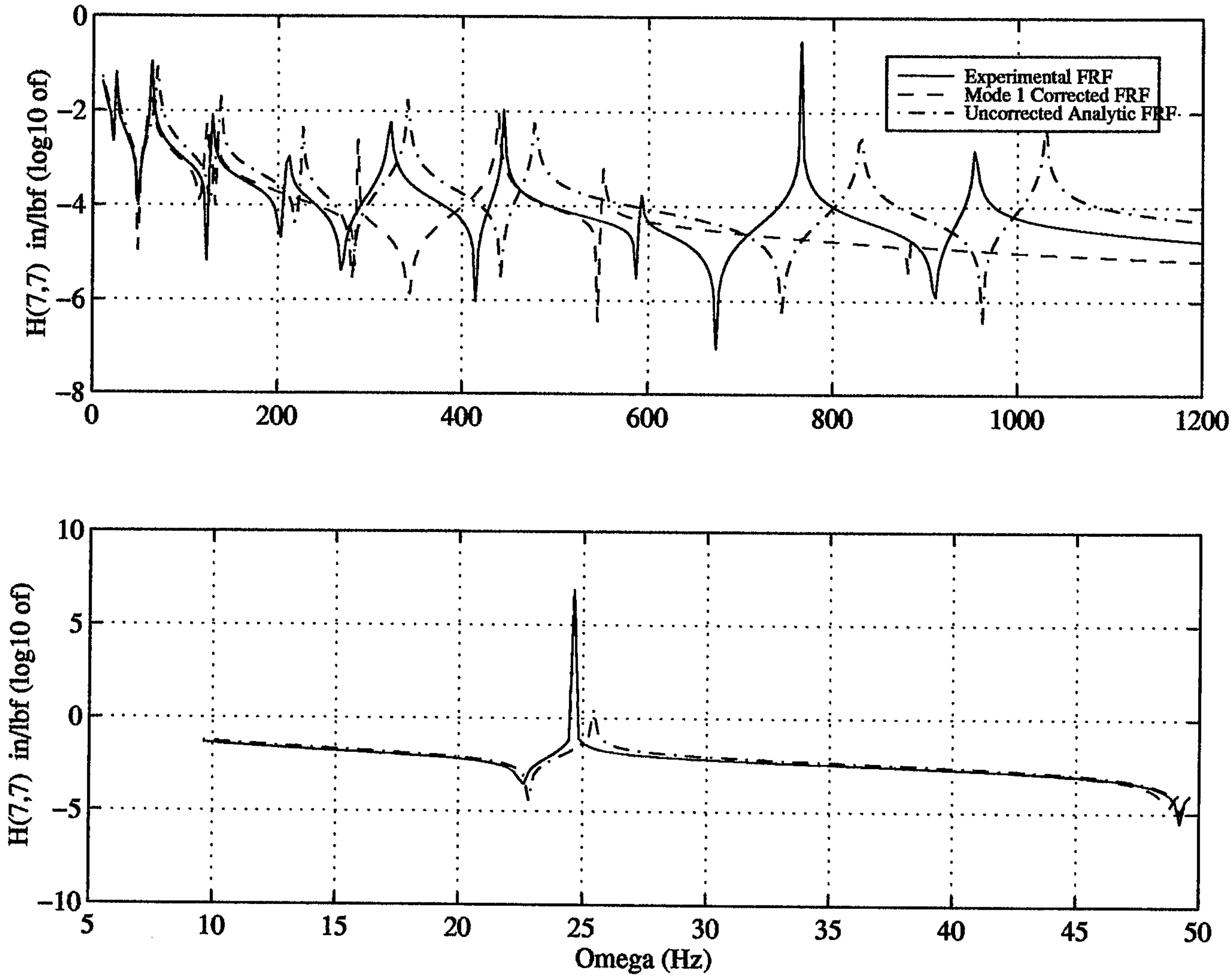

Figure 5- 1 Experimental and uncorrected Analytic FRFs vs single mode solution at mode 1 corrected FRF using a 3 point frequency sampling of a 1 Hz bandwidth.

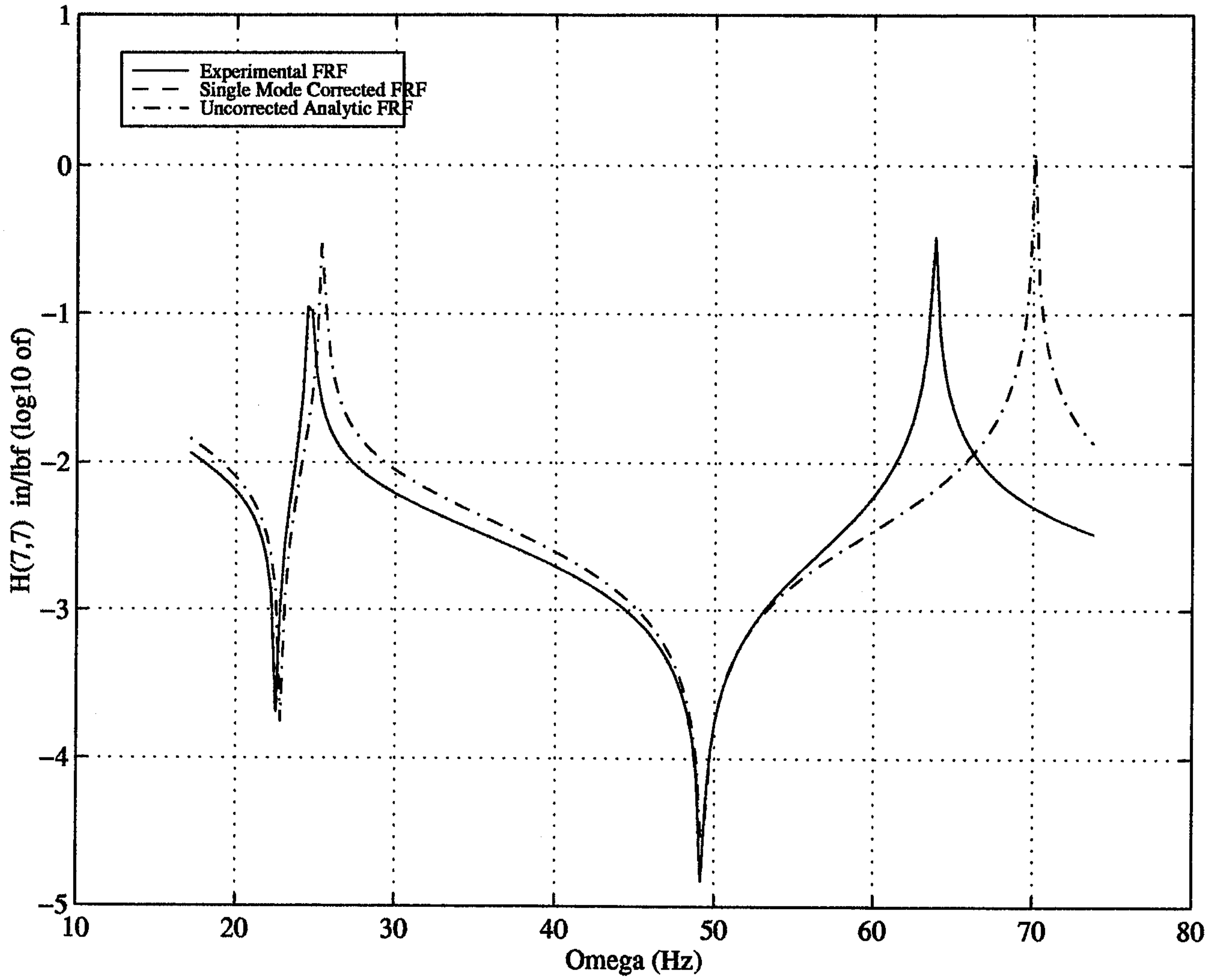

Figure 5- 2 Experimental and uncorrected analytic FRFs vs single mode corrected FRF using a 15 point frequency sampling of a bandwidth that includes modes 1 and 2.

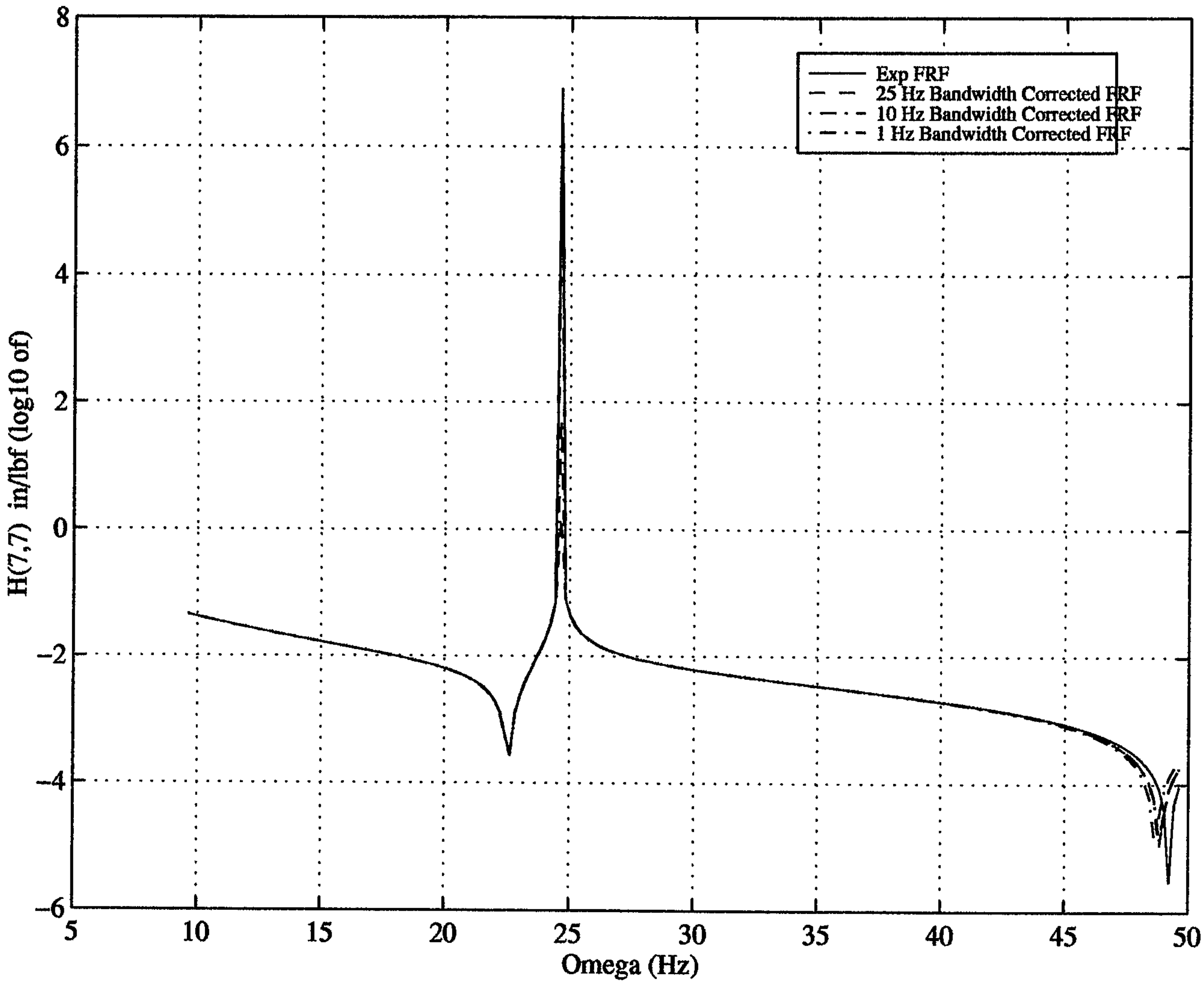

Figure 5- 3 Spatially incomplete experimental FRF vs single mode matrix solutions at mode 1 using 3 point frequency samplings of 25, 10, and 1 Hz bandwidths.

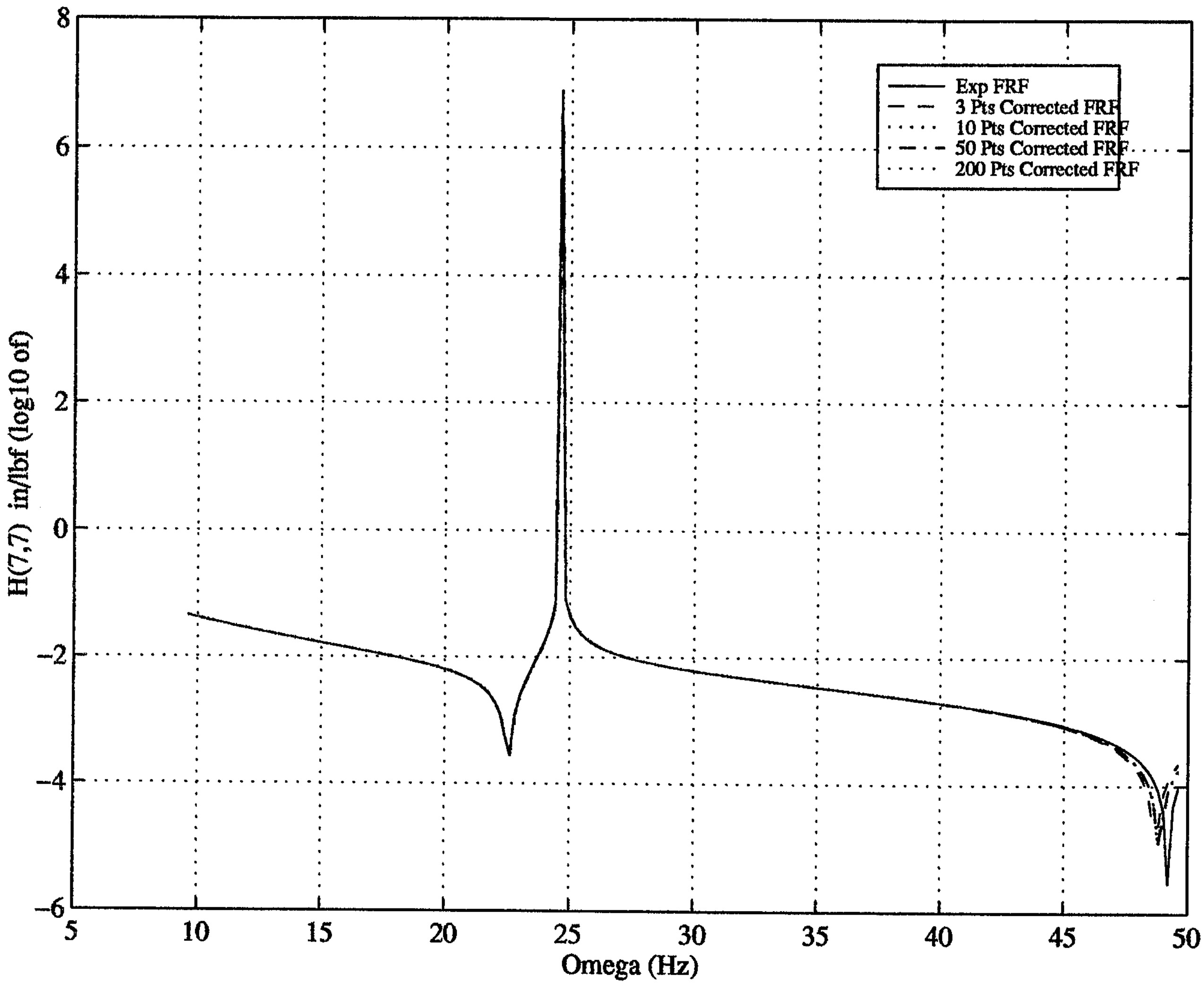

Figure 5- 4 Spatially incomplete experimental FRF vs single mode matrix solutions at mode 1 using 3, 10, 50, and 200 point frequency samplings of a 1 Hz bandwidth.

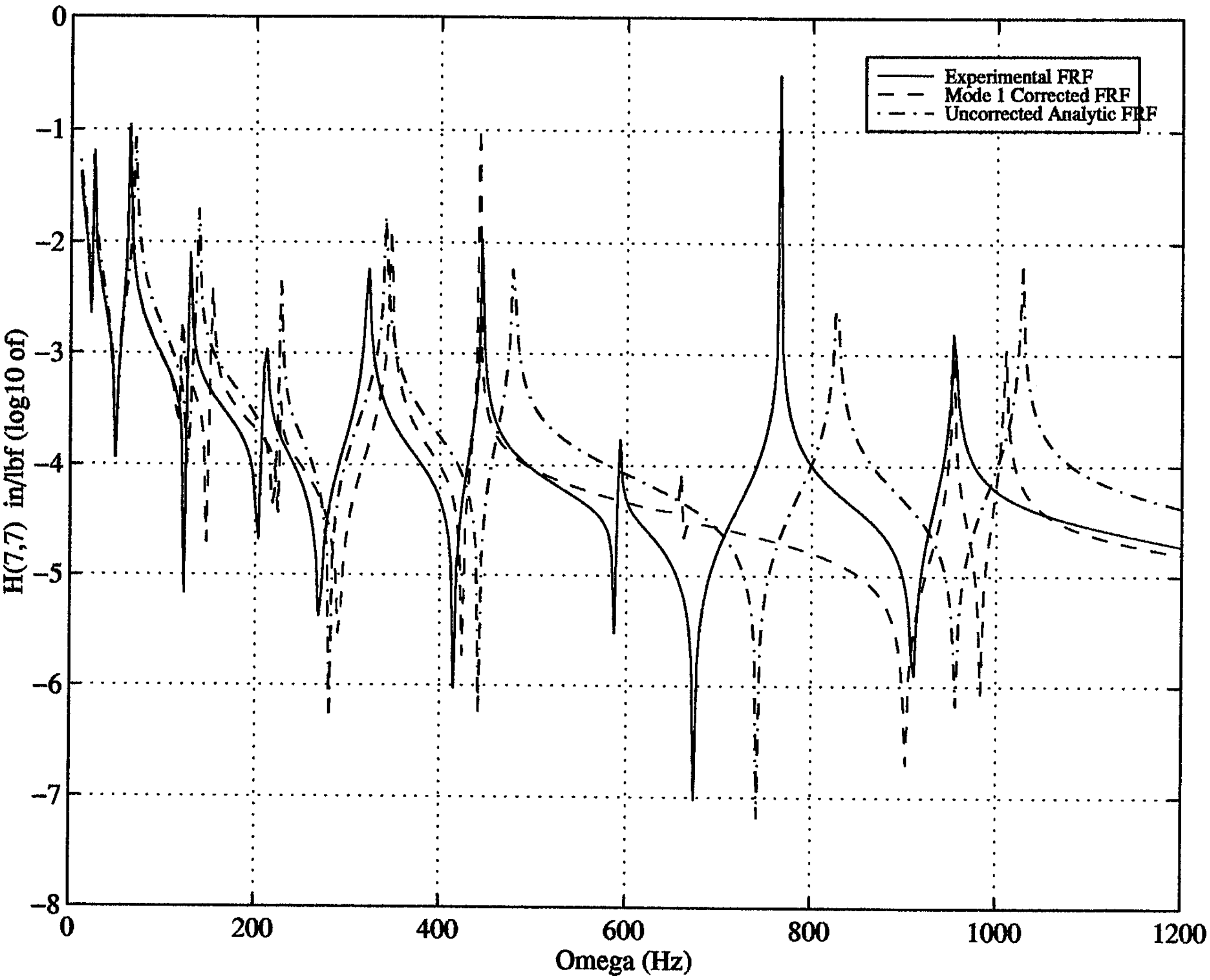

Figure 5- 5 Spatially complete experimental FRF vs single mode matrix solutions at mode 1 using a 3 point frequency samplings of a 1 Hz bandwidth.

B. SINGLE MODE INTEGRAL SOLUTIONS

In what follows we wish to employ the integral formulas of reference (5) to obtain a results similar to that of Equation (5.2). To accomplish this we shall need to recast the impedance equations of Chapter II in terms of velocity. We start with the equation of motion of a general 2nd order linear system

$$Kx + C\dot{x} + M\ddot{x} = f \tag{5.4a}$$

where

$$x = Xe^{j\Omega t} \tag{5.4b}$$

$$f = Fe^{j\Omega t} \tag{5.4c}$$

If we differentiate Equation (5.4b) we get that

$$\dot{x} = j\Omega Xe^{j\Omega t} = j\Omega x \tag{5.5}$$

hence

$$x = \frac{\dot{x}}{j\Omega} \tag{5.6}$$

differentiating Equation (5.4b) twice we get that

$$\ddot{x} = j\Omega\left(j\Omega Xe^{j\Omega t}\right) = j\Omega\dot{x} \tag{5.7}$$

substituting Equations (5.6)and (5.7) into Equation (5.4a) we get

$$K\frac{\dot{x}}{j\Omega} + C\dot{x} + Mj\Omega\dot{x} = f \tag{5.8}$$

We can rewrite equation (5.8) as

$$\left(\frac{K}{j\Omega} + C + Mj\Omega\right)\dot{x} = f \tag{5.9}$$

Following reference (5) we can take the Laplace transform of equation (5.9) and thus write the impedance of the linear system in terms of the complex Laplace parameter s as

$$Z(s) = Ms + C + \frac{K}{s} \tag{5-10}$$

In reference (5) the impedance matrix of a general system is represented as an infinite Laurent series about the orgin in powers of the complex Laplace parameter s as

$$Z(s) = \sum_{n=-\infty}^{\infty} A_n s^n = Lim_{n\to\infty}(A_n s^n + \cdots + A_1 s + A_0 + A_{-1}\frac{1}{s} + \cdots + A_{-n}\frac{1}{s^n}) \tag{5.11}$$

After defining a truncated Laurent series

$$\overline{Z}(s) = A_1 s + A_0 + A_{-1}\frac{1}{s} = \overline{M}s + \overline{C} + \frac{\overline{K}}{s} \tag{5.12}$$

which approximates the impedance matrix, an error function $E(s)$, and a cost J are defined as

$$E(s) = Z(s) - \overline{Z}(s) \tag{5.13}$$

and

$$J = \oint_{s=P} \|E(s)\|_E |W(s) \cdot ds| \tag{5.14}$$

where $W(s)$ is a complex valued weighting function, the subscript E denotes the euclidian norm of a matrix and the integration is performed over a prescribed path P in the complex plane. By setting the partial derivatives of J with respect to $\overline{M}$, $\overline{K}$, and $\overline{C}$ to zero the authors of reference (5) obtained expressions for matrices $\overline{M}$, $\overline{K}$, and $\overline{C}$ which approximate the stiffness, mass and damping matrices about a mode of the system. The expression for $\overline{M}$, $\overline{K}$, and $\overline{C}$ are as follows:

$$\overline{M} = \frac{1}{ab - c^2}\left[b\int_{\Omega_1}^{\Omega_2} \Omega Z_I(i\Omega)|W(i\Omega)|d\Omega - c\int_{\Omega_1}^{\Omega_2} \frac{1}{\Omega} Z_I(i\Omega)|W(i\Omega)|d\Omega \right] \tag{5-15a}$$

$$\overline{C} = \frac{1}{c}\int_{\Omega_1}^{\Omega_2} Z_R(i\Omega)|W(i\Omega)|d\Omega \tag{5-15b}$$

$$\overline{K} = \frac{1}{ab - c^2}\left[c\int_{\Omega_1}^{\Omega_2} \Omega Z_I(i\Omega)\left|W(i\Omega)\right| d\Omega - a\int_{\Omega_1}^{\Omega_2} \frac{1}{\Omega} Z_I(i\Omega)\left|W(i\Omega)\right| d\Omega \right]$$

(5-15c)

where

$$a = \int_{\Omega_1}^{\Omega_2} \Omega^2 \left|W(i\Omega)\right| d\Omega \tag{5-15d}$$

$$b = \int_{\Omega_1}^{\Omega_2} \frac{1}{\Omega^2} \left|W(i\Omega)\right| d\Omega \tag{5-15e}$$

$$c = \int_{\Omega_1}^{\Omega_2} \left|W(i\Omega)\right| d\Omega \tag{5-15f}$$

$$Z(i\Omega) = Z_R(i\Omega) + iZ_I(i\Omega) \tag{5-15g}$$

We will apply Equation (5.15) to ΔZ as defined by the SST and obtain matrices $\Delta\overline{K}$, $\Delta\overline{M}$, and $\Delta\overline{C}$ which will serve as correction matrices of the FRF of the system about a given mode of the system.

As an example, let us use Equation (5.15) to compute the (1,1) element of the correction matrix $\Delta\overline{C}$ at mode 1 (ω_1=25.21 Hz) for our spatially incomplete system. We take the weighting function to be *W(iΩ)=1/(iΩ)* , the path *P* is taken to be along the imaginary axis from 155.30 rads/sec to 161.58 rads/sec.

To compute this line integral we shall use the trapezoid rule with a three point frequency sampling

$$\Xi = \{155.30 rads/\sec, 158.44 rads/\sec, 161.58 rads/\sec\} \tag{5.16}$$

of the straight line path P (sampling frequency of π rads/sec). We first compute the value of the constant a of Equation (5.15d). Using vector notation we have that

$$\Omega(\Xi) = [155.30 rads/\sec, 158.44 rads/\sec, 161.58 rads/\sec] \tag{5.17a}$$

and

$$W(\Xi) = [0 - 0.0064j \sec/rads, 0 - 0.0063j \sec/rads, 0 - 0.0062j \sec/rads] \tag{5.17b}$$

hence the integrand is the vector

$$\Omega^2(\Xi)\cdot|W(\Xi)| = [155.30,158.44,161.58] \tag{5.18}$$

Using the trapezoid rule to compute the integral, we have that

$$a = \int_{155.30}^{161.58} \Omega^2(\Xi)|W(\Xi)|d\Omega = \frac{\pi}{2}\left(\frac{155.30}{2} + 158.44 + \frac{161.58}{2}\right) = 995.56\,{}^{rads^2}\!\big/_{\sec^2} \tag{5.19}$$

b=.00000158 sec²/rads² and *c=0.0397* are computed in a similar manner.

To actually compute the matrix $\Delta\overline{C}$, we use the SST to compute the matrices *ΔZ(155.30 rads/sec), ΔZ(155.44 rads/sec),* and *ΔZ(161.58 rads/sec)* for the points of the frequency sampling of the straight line path *P*. Using Equation (2.26) the matrix *ΔZ(155.30 rads/sec)* is seen to be

$$\Delta Z\left(155.44\,{}^{rads}\!\big/_{\sec}\right) = \begin{bmatrix} (0.64E-9)+0.02j & (-0.04E-8)-0.04j & (0.03E-7)+0.01j \\ (-0.4E-8)-0.04j & (-0.4E-9)-0.04j & (0.25E-7)-0.10j \\ (0.3E-8)+0.01j & (0.25E-7)-0.10j & (-0.24E-7)+0.15j \end{bmatrix} {}^{lbf}\!\big/_{in} \tag{5.20}$$

The integral in Equation (5.15b) is computed on an element by element basis. For the (1,1) element of Equation (5.15b), if we collect the (1,1) elements of the matrices *ΔZ(155.30 rads/sec), ΔZ(155.44 rads/sec),* and *ΔZ(161.58 rads/sec)* and use the weighting vector given in Equation (5.17b) the integrand of Equation (5.15b) is the vector

$$\Delta Z_R^{(1,1)}(\Xi)|W(\Xi)| = [0.0064 \quad 0.040 \quad -0.848][0.64 \quad 0.63 \quad 0.62](E-9)\,{}^{lbf}\!\big/_{in}\,{}^{\sec}\!\big/_{rads} \tag{5.21a}$$

$$\Delta Z_R^{(1,1)}(\Xi)|W(\Xi)| = [0.0041 \quad 0.2885 \quad 0.0036](E-9)\,{}^{lbf}\!\big/_{in}\,{}^{\sec}\!\big/_{rads} \tag{5.21b}$$

Computing the integral we have

$$\Delta\overline{C}(1,1) = \frac{1}{c}\int_{155.30}^{161.58} \Delta Z_R^{(1,1)}(\Xi)|W(\Xi)|d\Omega = \frac{1}{0.0397}\frac{\pi}{2}\left(\frac{0.0041}{2} + 0.2885 + \frac{0.0036}{2}\right)(E-9)\,{}^{lbf}\!\big/_{in} \tag{5.22a}$$

$$\Delta\overline{C}(1,1) = 0.0232E-6\,{}^{lbf}\!\big/_{in} \tag{5.22b}$$

Performing the above procedures for the remaining elements of Equation (5.15b) we get

$$\Delta\overline{C} = \begin{bmatrix} 0.0232 & -0.0443 & 0.0300 \\ -0.0443 & 0.1278 & -0.0914 \\ 0.0300 & -0.0914 & 0.0645 \end{bmatrix} (E-6)\, {}^{lbf}\!/_{in} \tag{5.23a}$$

In a similar manner we have

$$\Delta\overline{M} = \begin{bmatrix} -0.0001 & -0.0001 & 0.0002 \\ -0.0001 & 0.0015 & -0.0020 \\ 0.0002 & -0.0020 & 0.0027 \end{bmatrix} lbm \tag{5.23b}$$

$$\Delta\overline{K} = \begin{bmatrix} -6.0476 & 4.5347 & 3.6757 \\ 4.5347 & 51.2196 & -71.7061 \\ 3.6757 & -71.7061 & 85.4417 \end{bmatrix} {}^{lbf}\!/_{in} \tag{5.23c}$$

Figure 5-6 is a comparison plot of experimental and uncorrected analytic FRFs versus single mode integral solutions at mode 1 using a frequency sampling of 3 points equally spaced over bandwidths of 25, 10, and 1 Hz. As with the matrix solutions, the integral solutions are sensitive to the length of the bandwidth over which the solutions are computed with smaller bandwidths yielding more accurate solutions. Figure 5-7 is a comparison plot of experimental and uncorrected analytic FRFs versus single mode integral solutions at mode 1 over a 1 Hz bandwidth using frequency samplings of 3,10,50, and 200 points from the bandwidth. As with the matrix solutions, the integral solutions are also insensitive to the sampling frequency.

Figures 5-8, 5-9, and 5-10 are comparison plots of matrix and integral solutions at mode 1 computed over 25, 10, and 1 Hz bandwidths respectively, using three point frequency samplings. These three plots show that for these frequency sampling, the matrix solutions are more accurate than the integral solutions. We must note that we have used the trapezoid rule for computation of the integrals. It is expected that better accuracy would be achieved from the integral solutions if a higher order integration method such as Simpson's rule were used.

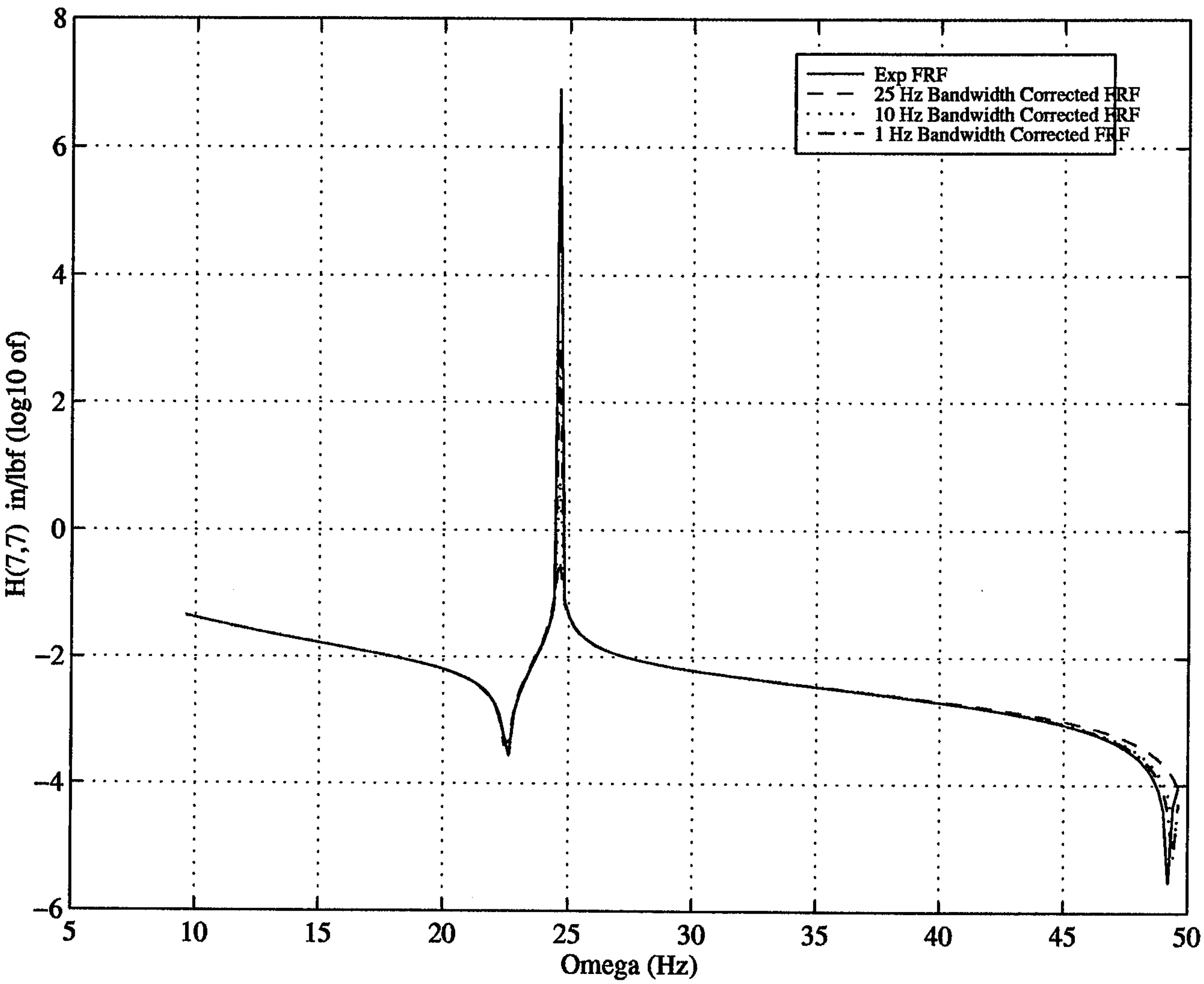

Figure 5- 6 Spatially incomplete experimental FRF vs single mode integral solutions at mode 1 using 3 point frequency samplings of 25, 10, and 1 Hz bandwidths.

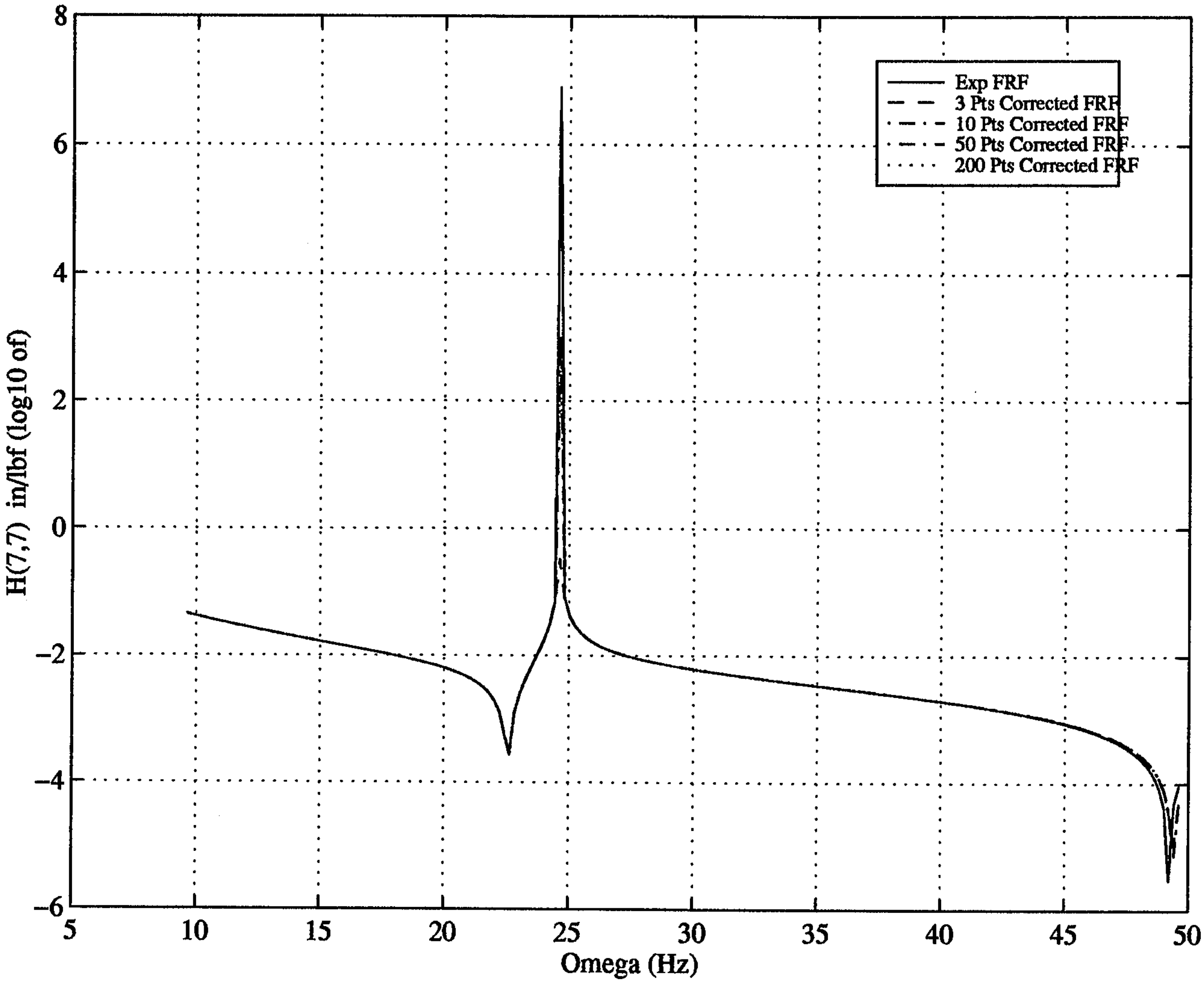

Figure 5-7 Spatially incomplete experimental FRF vs single mode integral solutions at mode 1 using 3, 10, 50 , and 200 point frequency samplings of a 1 Hz bandwidth.

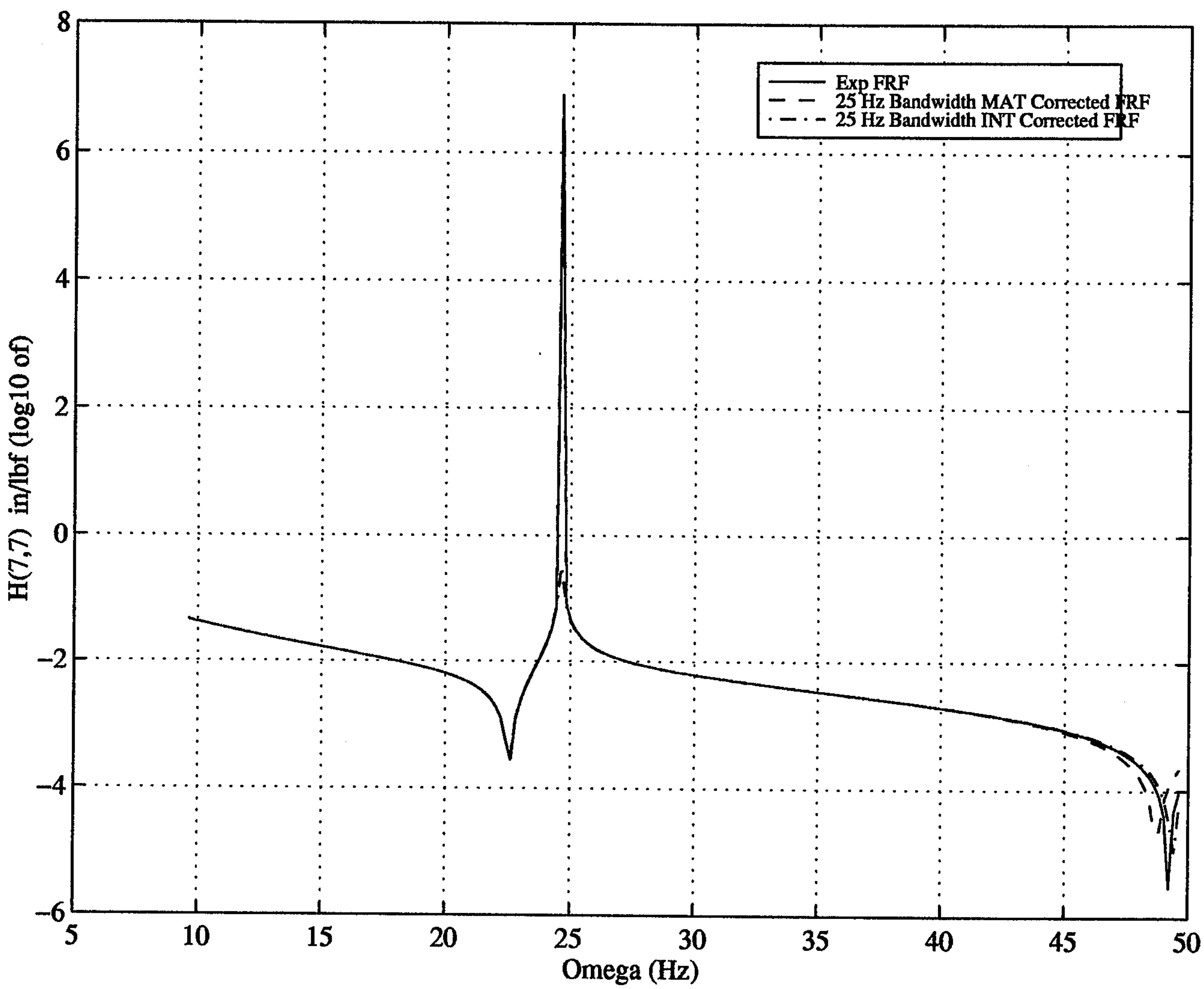

Figure 5- 8 Experimental FRF vs single mode integral and matrix solutions at mode 1 using a 25 Hz bandwidth with a 3 point frequency sampling.

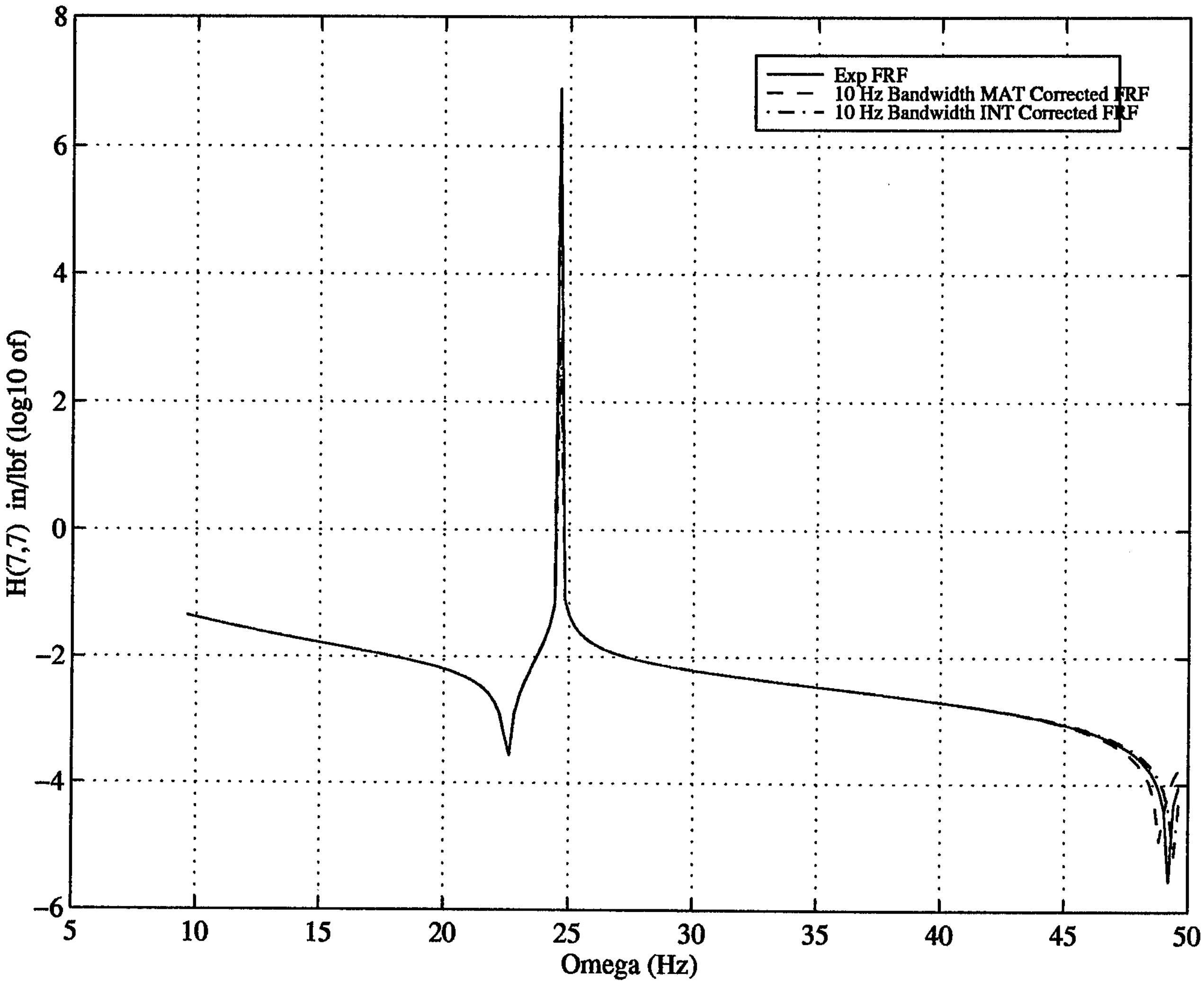

Figure 5- 9 Experimental FRF vs single mode integral and matrix solutions at mode 1 using a 10 Hz bandwidth with a 3 point frequency sampling.

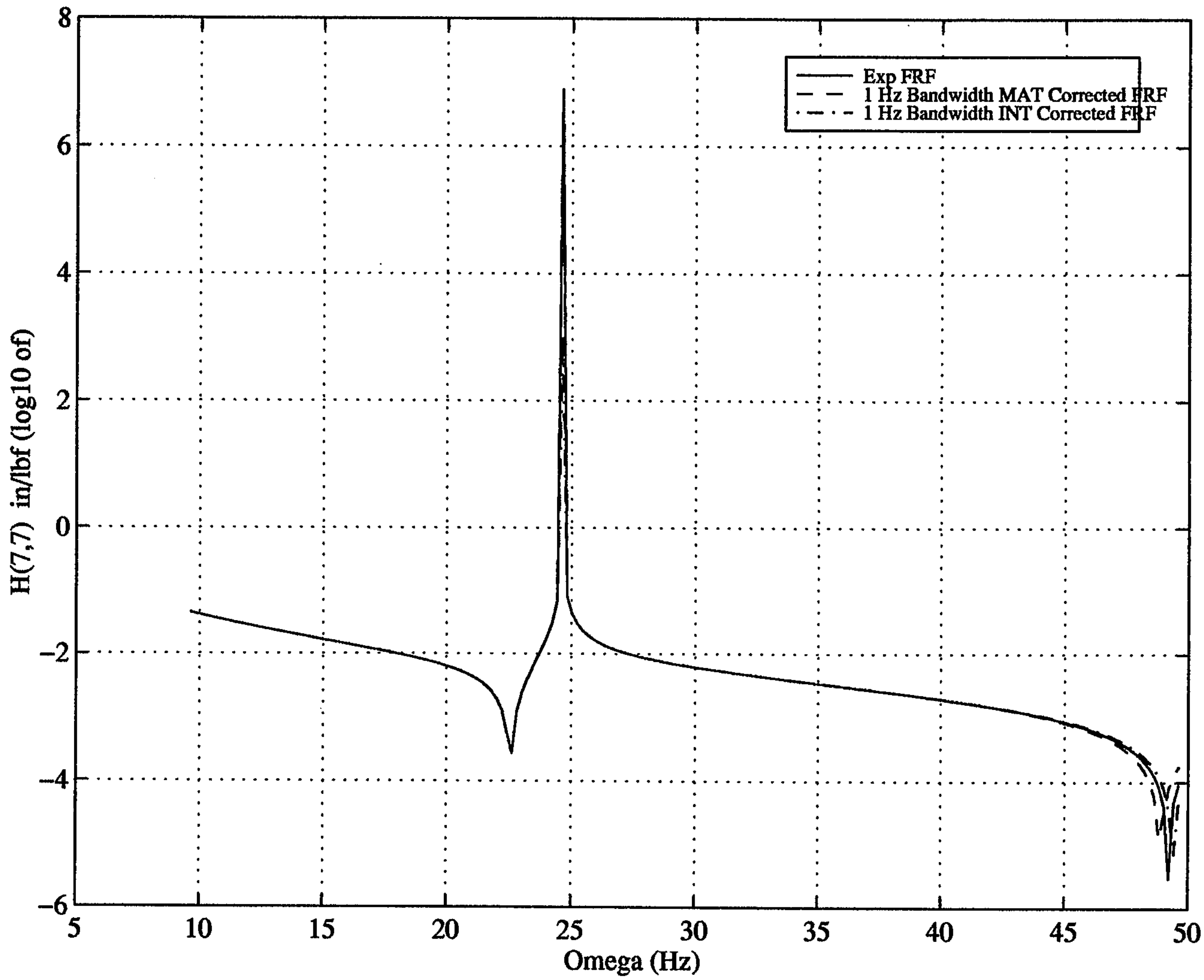

Figure 5- 10 Experimental FRF vs single mode integral and matrix solutions at mode 1 using a 1 Hz bandwidth with a 3 point frequency sampling.

VI. MULTIPLE MODE SOLUTIONS

A. MULTIPLE MODE MATRIX SOLUTIONS

We wish to extend the results of Chapter V to multiple modes of the system under consideration. To accomplish this we shall simply extend Equation (5.1) over multiple frequency samplings. For a set of system modes $\{\omega_1, \omega_2, \ldots, \omega_n\}$, let $\Xi(\omega_i)=\{\Omega_{i1}, \Omega_{i2}, \ldots, \Omega_{ik}\}$ be a frequency sampling of a bandwidth $[\Omega_{il}, \Omega_{iu}]$ about ω_i for i=1,2,...,n. We shall apply Equation (2.30) to the frequency sampling

$$\Xi = \bigcup_{i=1}^{n} \Xi(\omega_i) \tag{6.1}$$

which is the set theoretic union of the sets $\Xi(\omega_i)$ for i=1,2,...,n. If, for simplicy, we restrict ourselves to equally sized frequency samplings we obtain a set on nk equation in three unknowns

$$\begin{bmatrix} \Delta Z_c(\Omega_{11}) \\ \vdots \\ \Delta Z_c(\Omega_{ij}) \\ \vdots \\ \Delta Z_c(\Omega_{nk}) \end{bmatrix} = \begin{bmatrix} I & -\Omega_{11}^2 I & j\Omega_{11} I \\ \vdots & \vdots & \vdots \\ I & -\Omega_{ij}^2 I & j\Omega_{ij} I \\ \vdots & \vdots & \vdots \\ I & -\Omega_{nk}^2 I & j\Omega_{nk} I \end{bmatrix} \begin{bmatrix} \Delta K_c^{\Xi} \\ \Delta M_c^{\Xi} \\ \Delta C_c^{\Xi} \end{bmatrix} \tag{6.2}$$

One can solve Equation (6.2) for $\begin{bmatrix} \Delta K_c^{\Xi} \\ \Delta M_c^{\Xi} \\ \Delta C_c^{\Xi} \end{bmatrix}$ as we have done in Chapter V using the familiar MATLAB puedoinverse function but better results are obtained if we weight the equations as discussed in [Ref. 6]. A good weighting is to assign the weight ω_i to those equations associated with the frequency sampling $\Xi(\omega_i)$. Alternately we could assign the weight $1/\omega_i$ to those equations associated with the frequency sampling $\Xi(\omega_i)$. The ω_i weighting results in the solution being more accurate at the lower modes while the $1/\omega_i$ weighting gives better accuracy at the higher modes.

We note that the solution to Equation (6.2) approximately corrects the analytic FRF in the sense of Equation (5.1) when the matrices $\Delta Z(\Omega_{ij})$ as given by Equation (2.26),

are computed over a 'good' error set, C_{err}. In general, due to the smearing that occurs during reduction of the full analytic system to the reduced analytic system, the set C_{err} is not the same set as the intersection of the spatially complete error set and the A set. It is in general a larger set than this intersection and the size is a function of the type of error, i.e., mass, damping, or stiffness error and the locations of the errors. Figure 6-1 is a comparison plot of experimental and uncorrected analytic FRFs versus the multiple mode matrix solution corrected FRF with the solution having been computed over modes 1 and 2 using a 1 Hz bandwidth about each mode with a 3 point frequency sampling over the bandwidth at each of the modes. Figure 6-2 is a comparison plot using a matrix solution computed over mode 1 through 4 again using a 1 Hz bandwidth about each mode with a 3 point frequency sampling over the bandwidth at each of the modes.

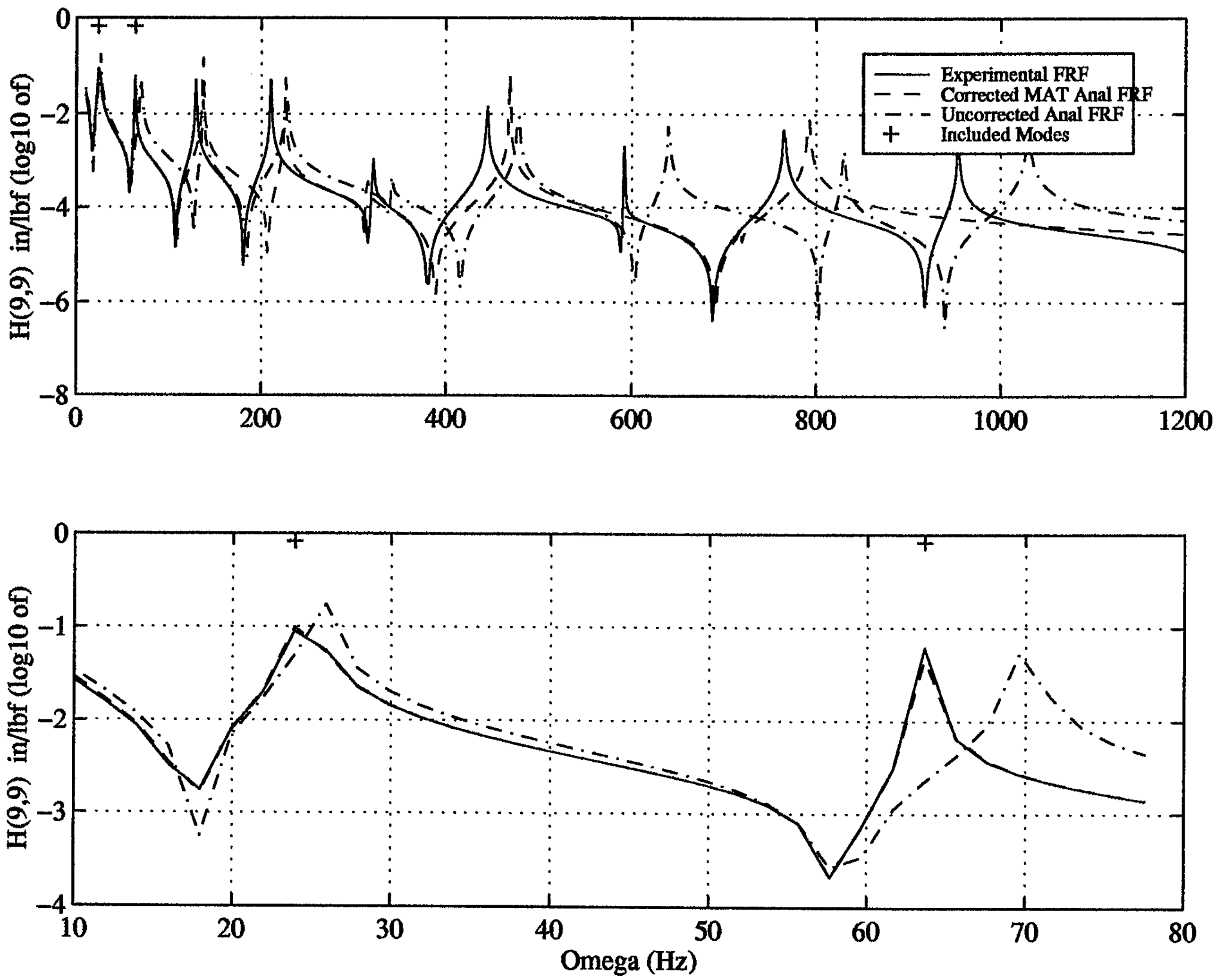

Figure 6- 1 Experimental FRF vs multiple mode matrix solutions at modes 1 and 2 using a 1 Hz bandwidth with a 3 point frequency samplings at each mode.

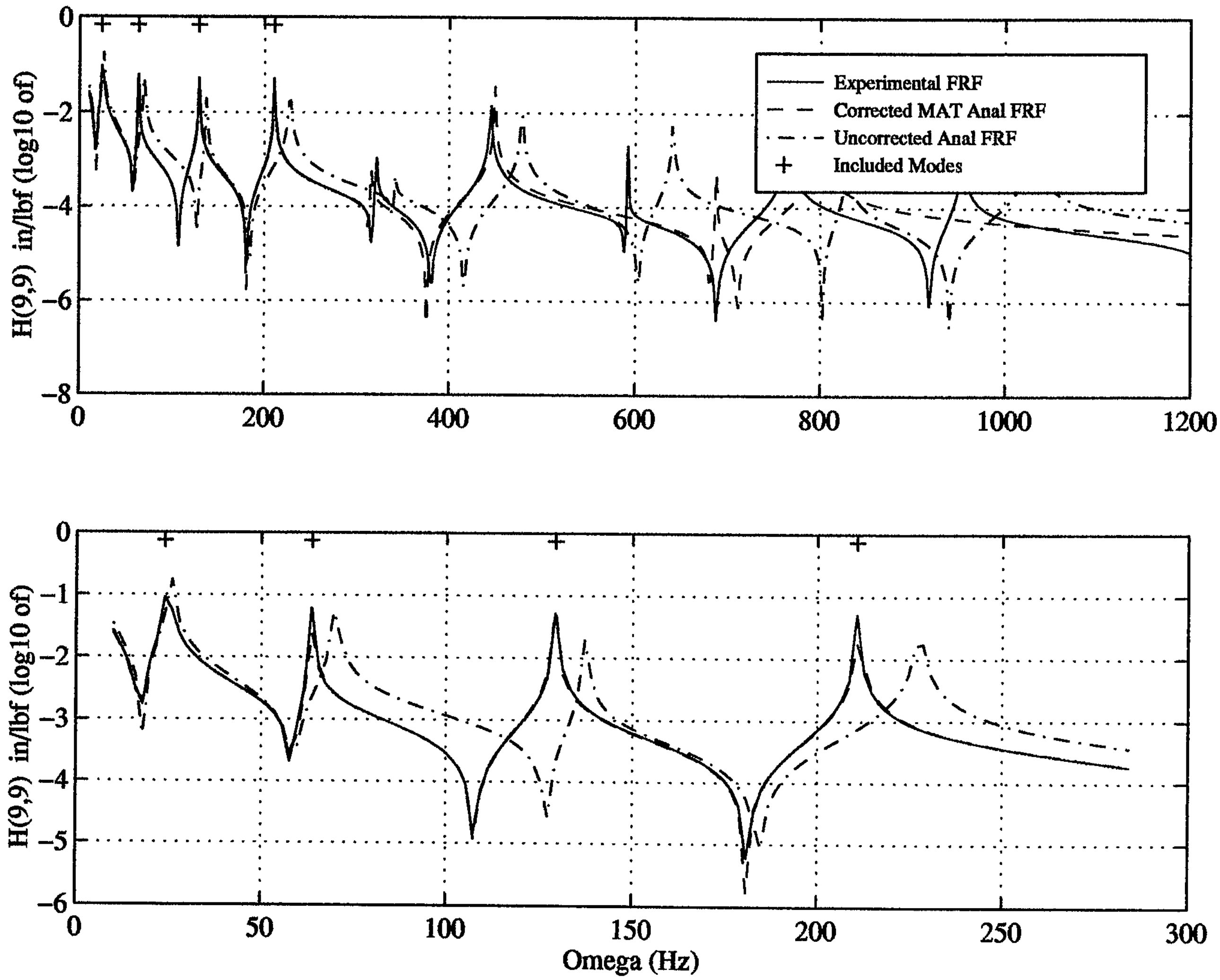

Figure 6- 2 Experimental FRF vs multiple mode matrix solutions at modes 1 through 4 using a 1 Hz bandwidth with a 3 point frequency samplings at each mode.

B. MULTIPLE MODE INTEGRAL SOLUTIONS

We wish to extend single mode integral solutions to multiple modes of the system. As with the multiple mode matrix solutions we simply take as the path of integration a path which is the set theoretic union of suitable paths at each of the modes under consideration. As with the matrix solutions one can chose the weighting function W(s) to be $1/(\Omega)$ or Ω to achieve lower or higher mode accuracy. As stated in the previous section it is required that a 'good' error set is known. Figure 6-3 is a comparison of experimental and uncorrected FRF versus multiple mode integral solution corrected FRF with the solution having been computed over modes 1 and 2 using a 1 Hz bandwidth at each mode with a 3 point frequency sampling over the bandwidth at each of the modes. Figure 6-4 is a comparison using a solution computed over mode 1 through 4 again using a 1 Hz bandwidth at each mode with a 3 point frequency sampling over the bandwidth at each of the modes. Figure 6-5 is a comparison plot of the multiple mode matrix and integral solutions over modes 1 through mode 4 using a 1 Hz bandwidth at each mode with a 3 point frequency sampling over the bandwidth at each of the modes.

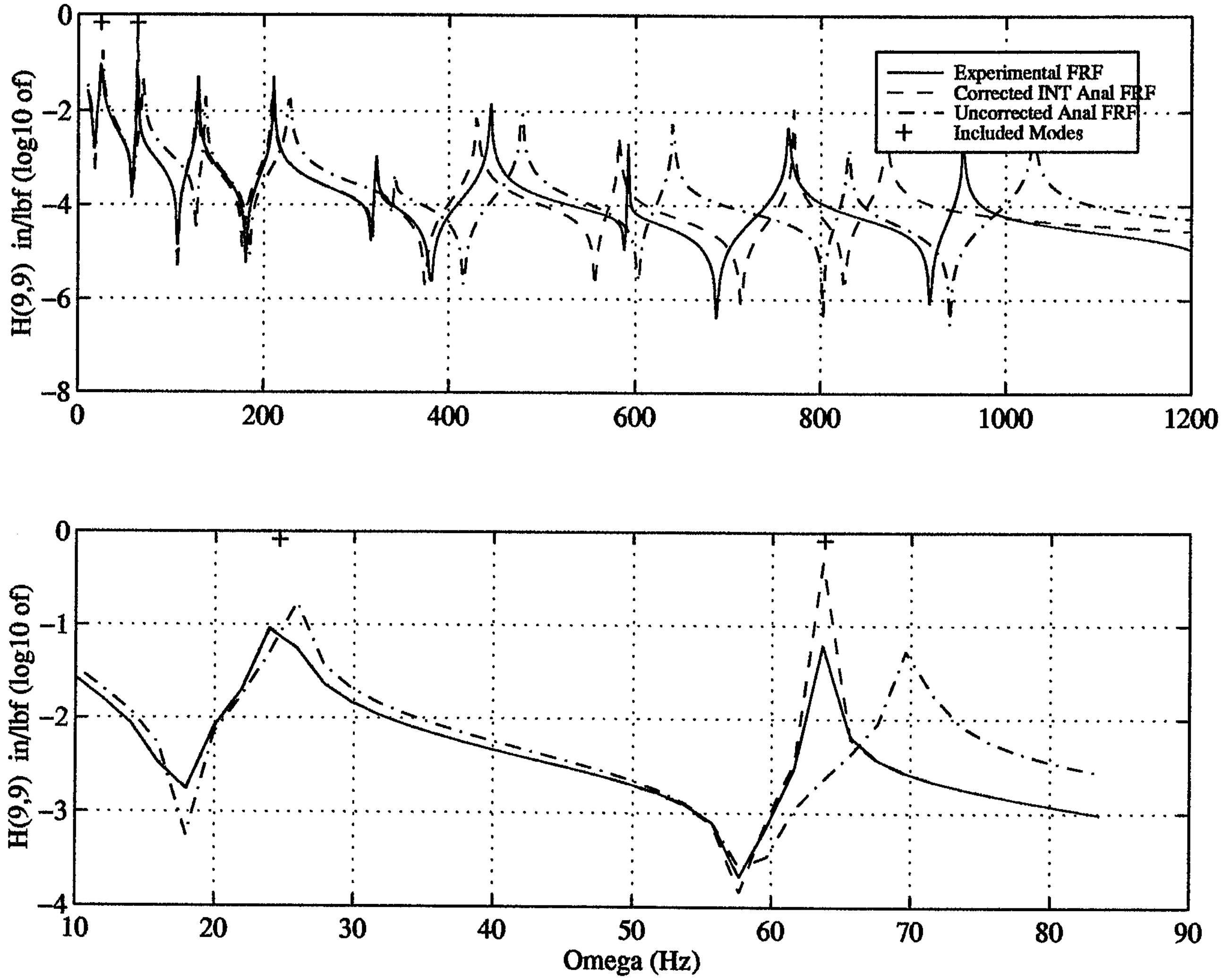

Figure 6- 3 Experimental FRF vs multiple mode integral solutions at modes 1 and 2 using a 1 Hz bandwidth with a 3 point frequency samplings at each mode.

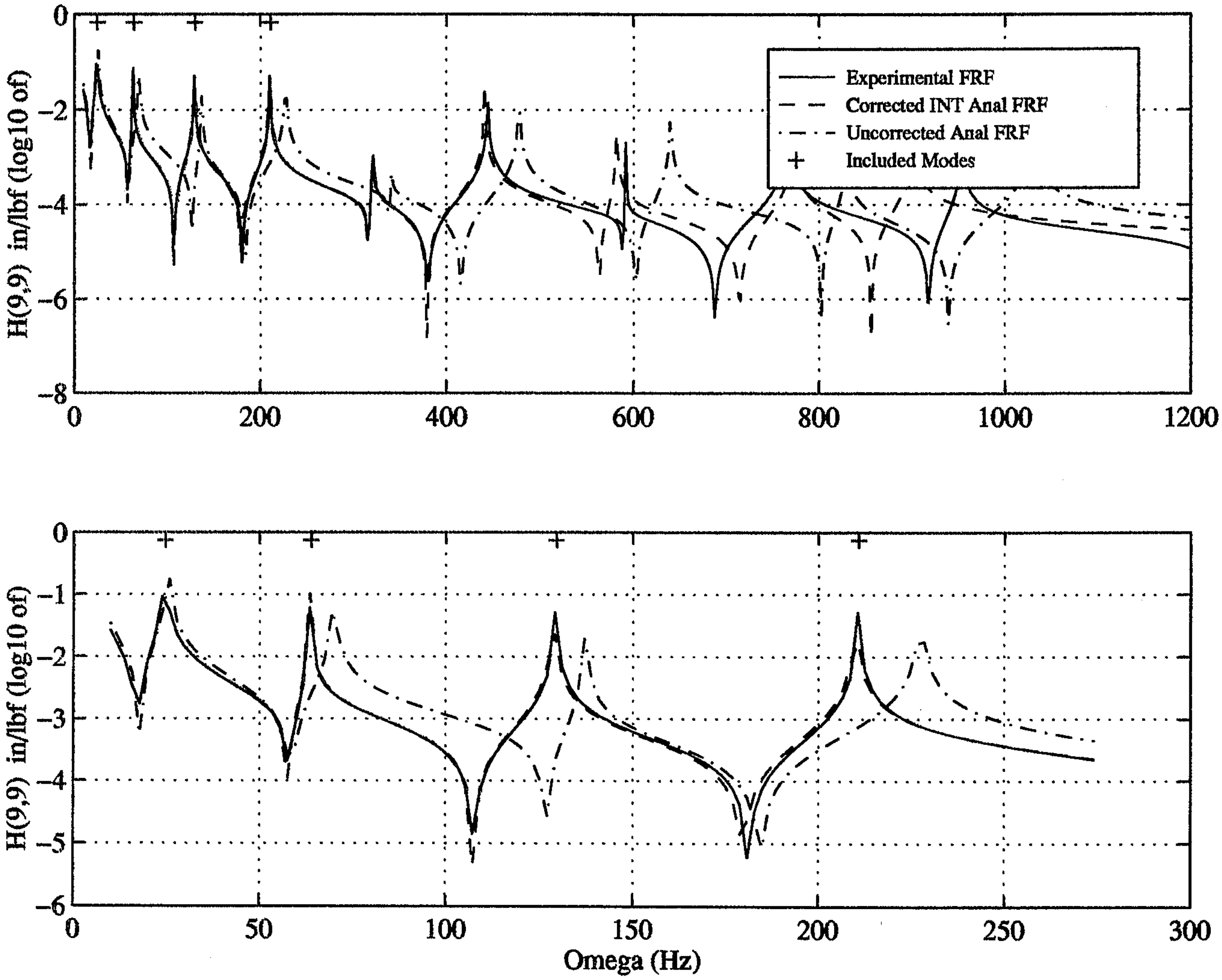

Figure 6- 4 Experimental FRF vs multiple mode integral solution at modes 1 through 4 using a 1 Hz bandwidth with a 3 point frequency samplings at each mode.

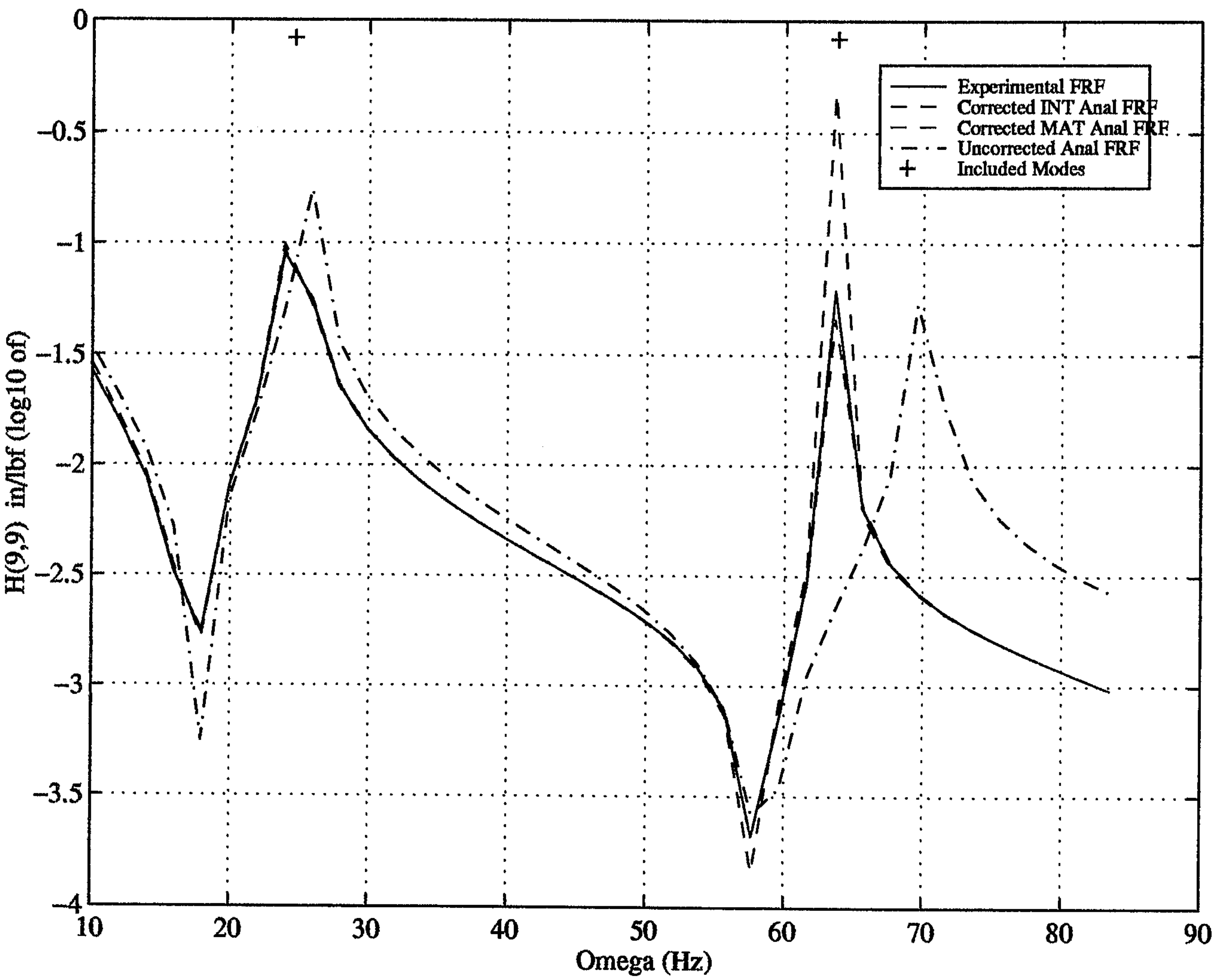

Figure 6- 5 Experimental FRF vs multiple mode matrix and integral solutions at modes 1 and 2 using a 1 Hz bandwidth with a 3 point frequency samplings at each mode.

C. SINGLE POINT MULTIPLE MODE SOLUTIONS

The results of sections A and B can be performed using partitions consisting of a single point. Figure 6-6 shows the results of performing a multiple mode matrix solution over the first three modes of our spatially incomplete beam using a single point partition at each of the modes under consideration. Figure 6-7 shows the results of performing a multiple point integral solution over the first three modes of our spatially incomplete beam using a single point partition at each of the modes under consideration. Figure 6-8 compares the multiple mode matrix and integral solution computed over the first three modes of our spatially incomplete beam with the analytic FRF. In figure 6-9 we compare a multiple mode matrix solution computed over the first three modes of a spatially complete beam using a single point partition at each of the modes with the beam's spatially complete analytic FRF. In figure 6-10 we compare a multiple mode integral solution computed over the first three modes of a spatially complete beam using a single point partition at each of the modes with the beam's spatially complete analytic FRF. Figure 6-11 compares multiple mode matrix and intgral solution computed over the first three modes of a spatially complete beam with the beam's spatially complete analytic FRF. Figures 6-9 and 6-10 show that the single point multiple mode solutions are exact.

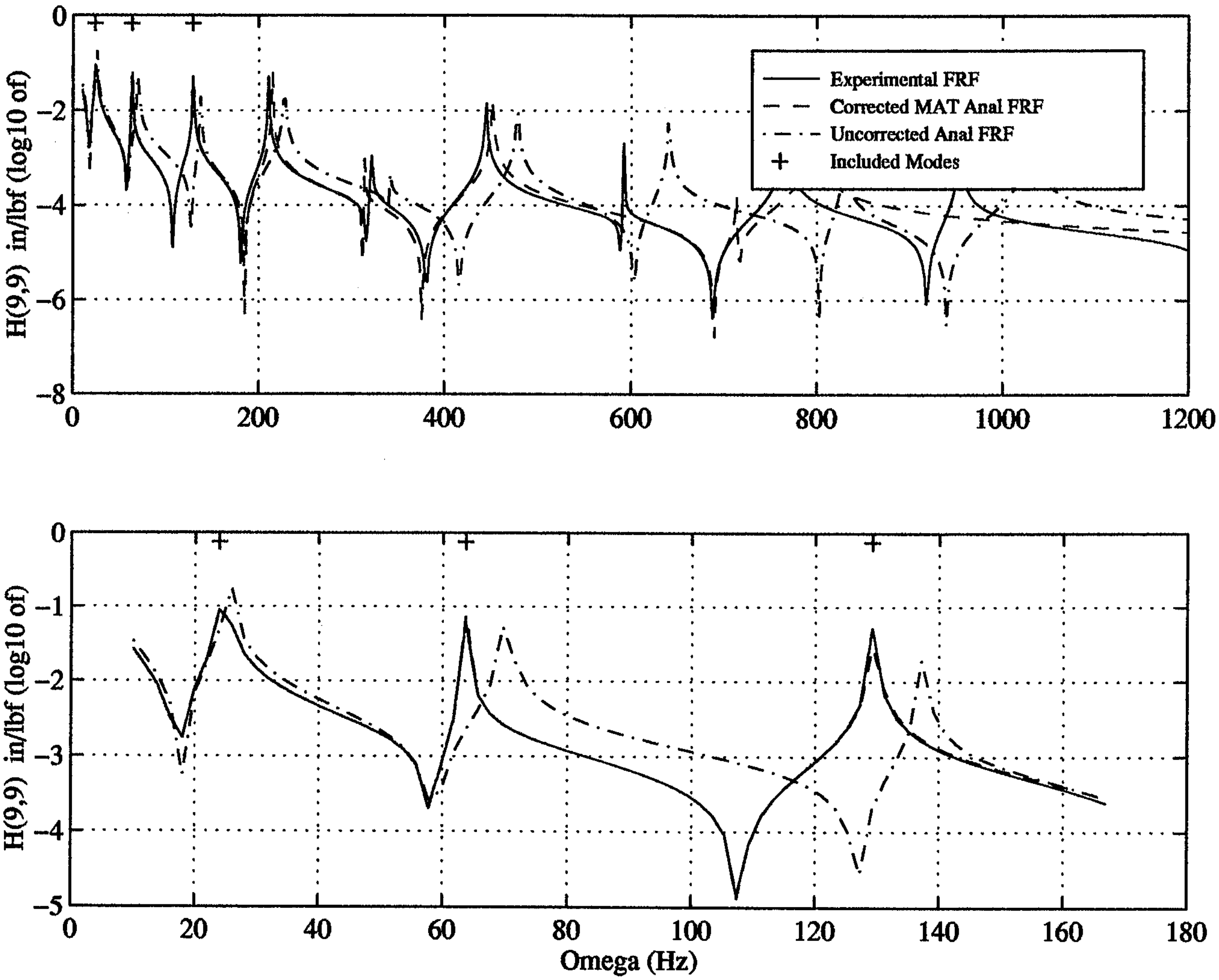

Figure 6- 6 Experimental FRF vs multiple mode matrix solution computed at modes 1 through 3 using a 1 Hz bandwidth with a single point frequency samplings at each mode for a spatially incomplete beam.

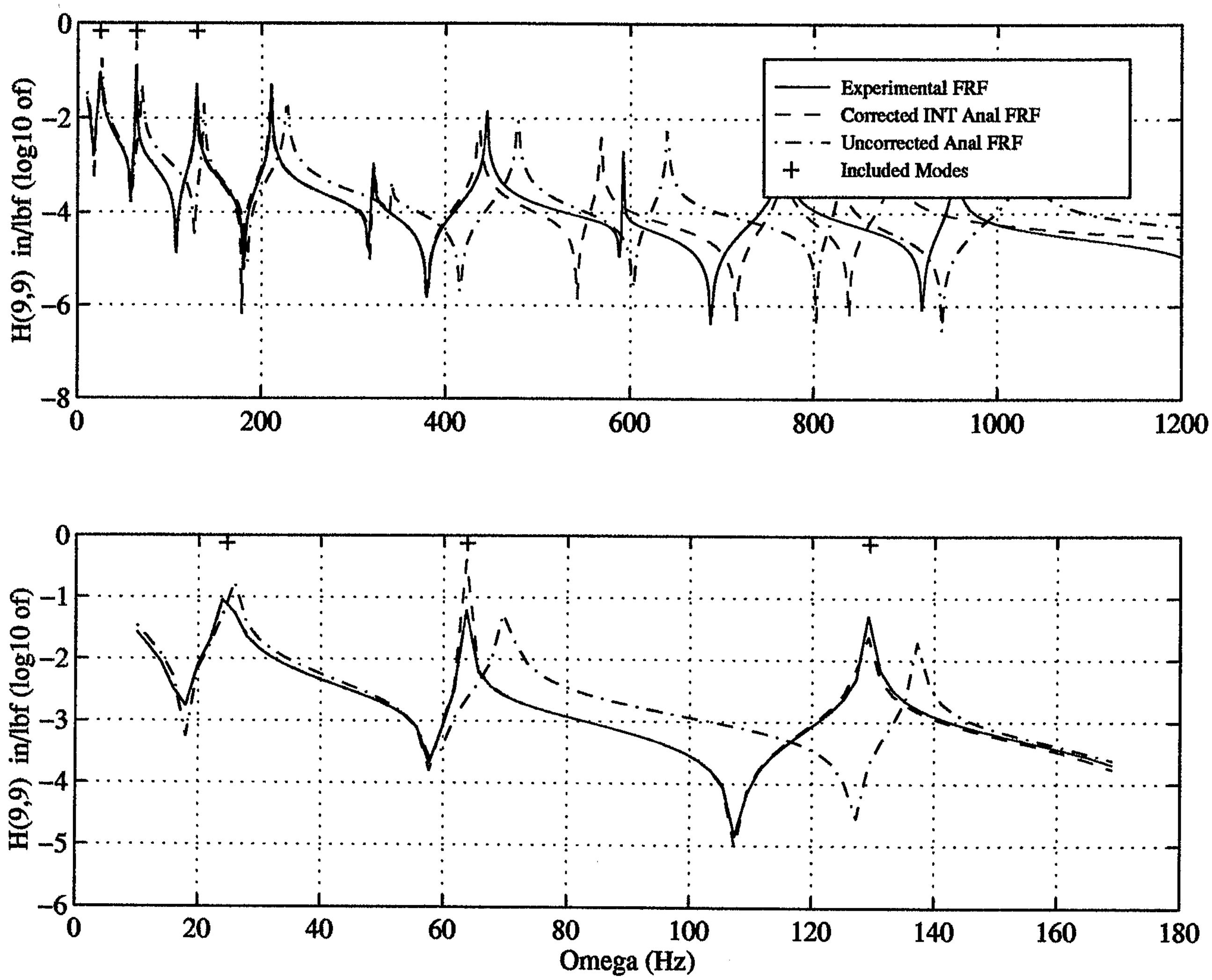

Figure 6- 7 Experimental FRF vs multiple mode integral solution computed at modes 1 through 3 using a 1 Hz bandwidth with a single point frequency samplings at each mode for a spatially incomplete beam.

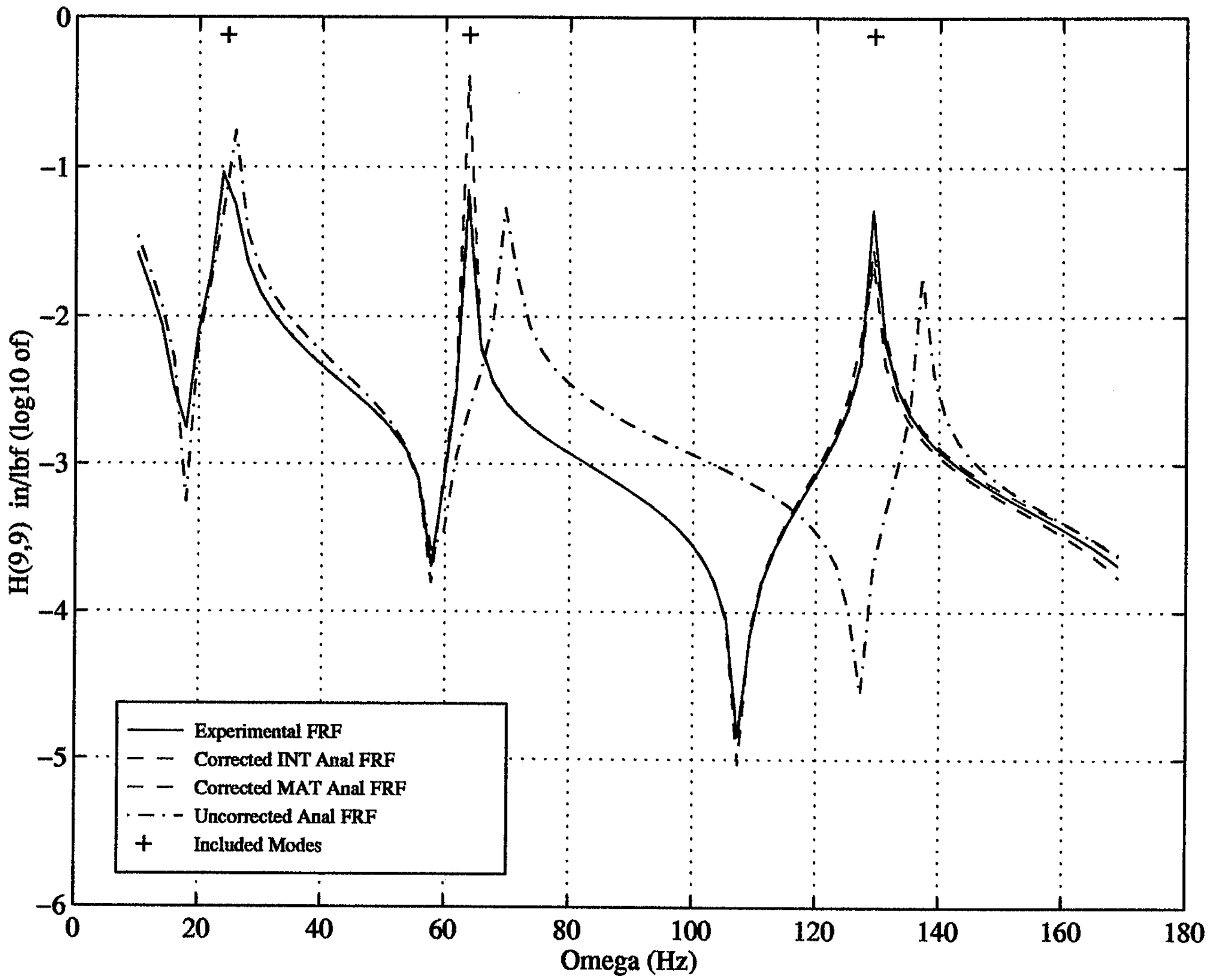

Figure 6- 8 Experimental FRF vs multiple mode matrix and integral solutions computed at modes 1 through 3 using a 1 Hz bandwidth with single point frequency samplings at each mode for a spatially incomplete beam.

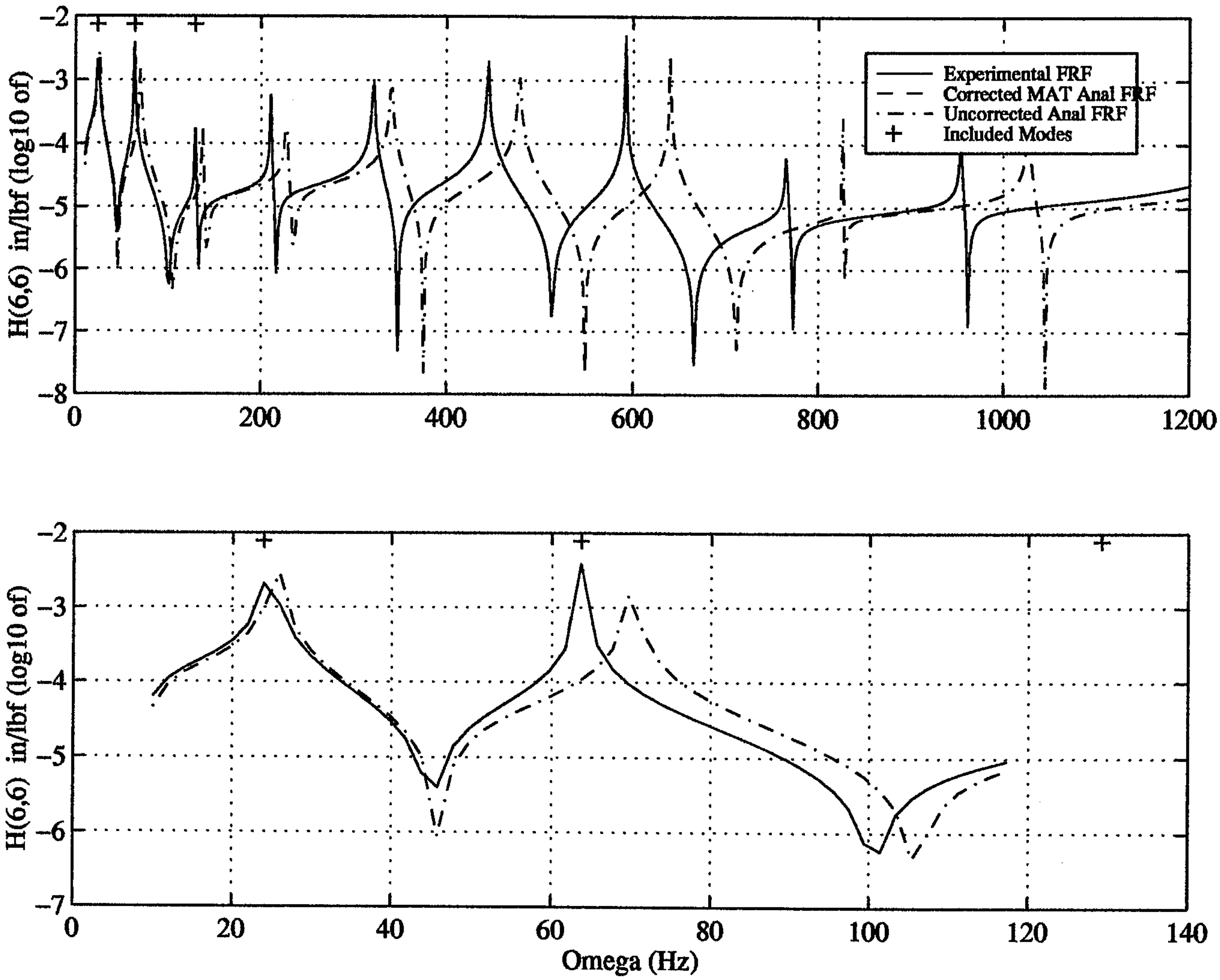

Figure 6- 9 Experimental FRF vs multiple mode matrix solution computed at modes 1 through 3 using a 1 Hz bandwidth with a single point frequency samplings at each mode for a spatially complete beam.

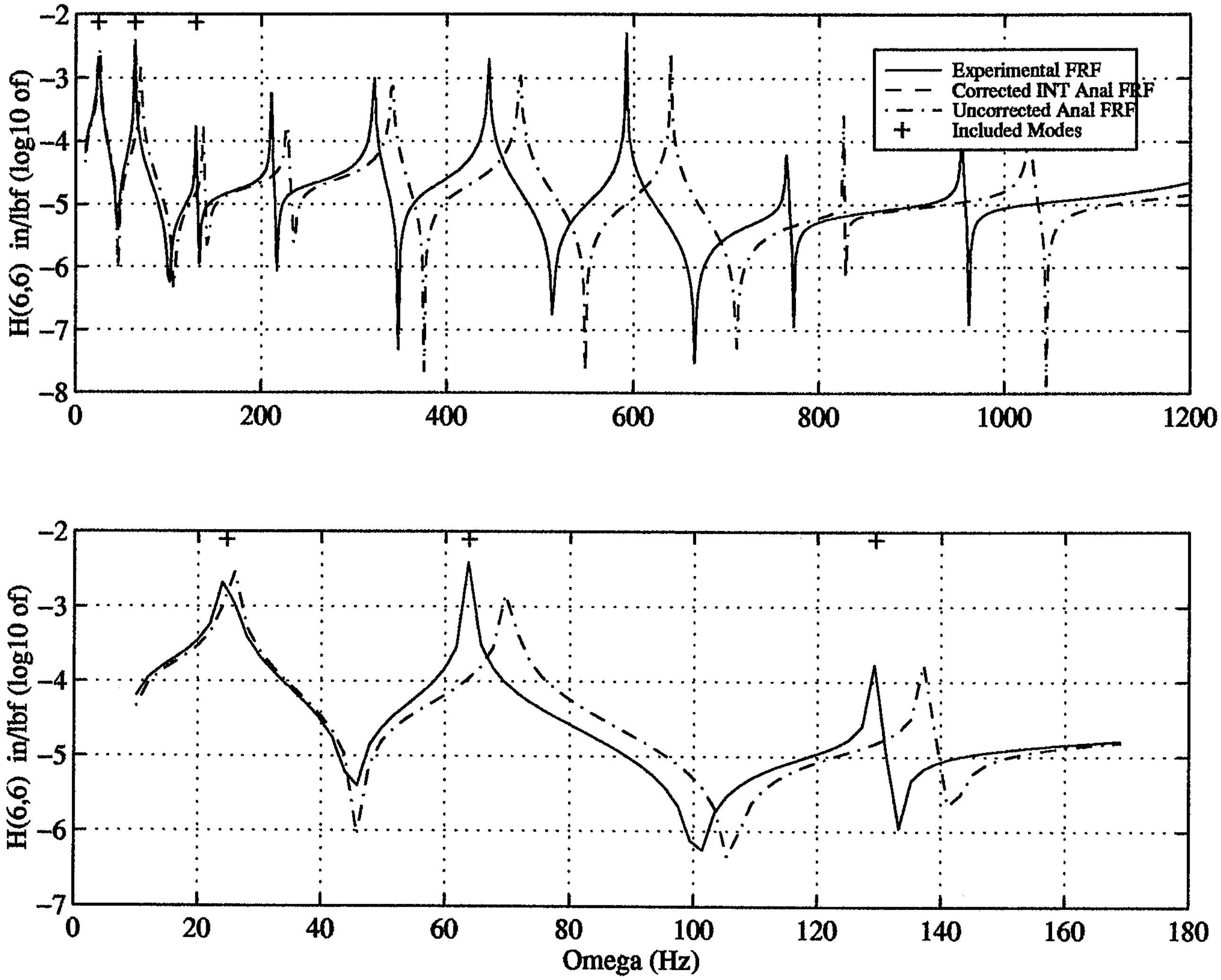

Figure 6- 10 Experimental FRF vs multiple mode integral solution computed at modes 1 through 3 using a 1 Hz bandwidth with a single point frequency samplings at each mode for a spatially complete beam.

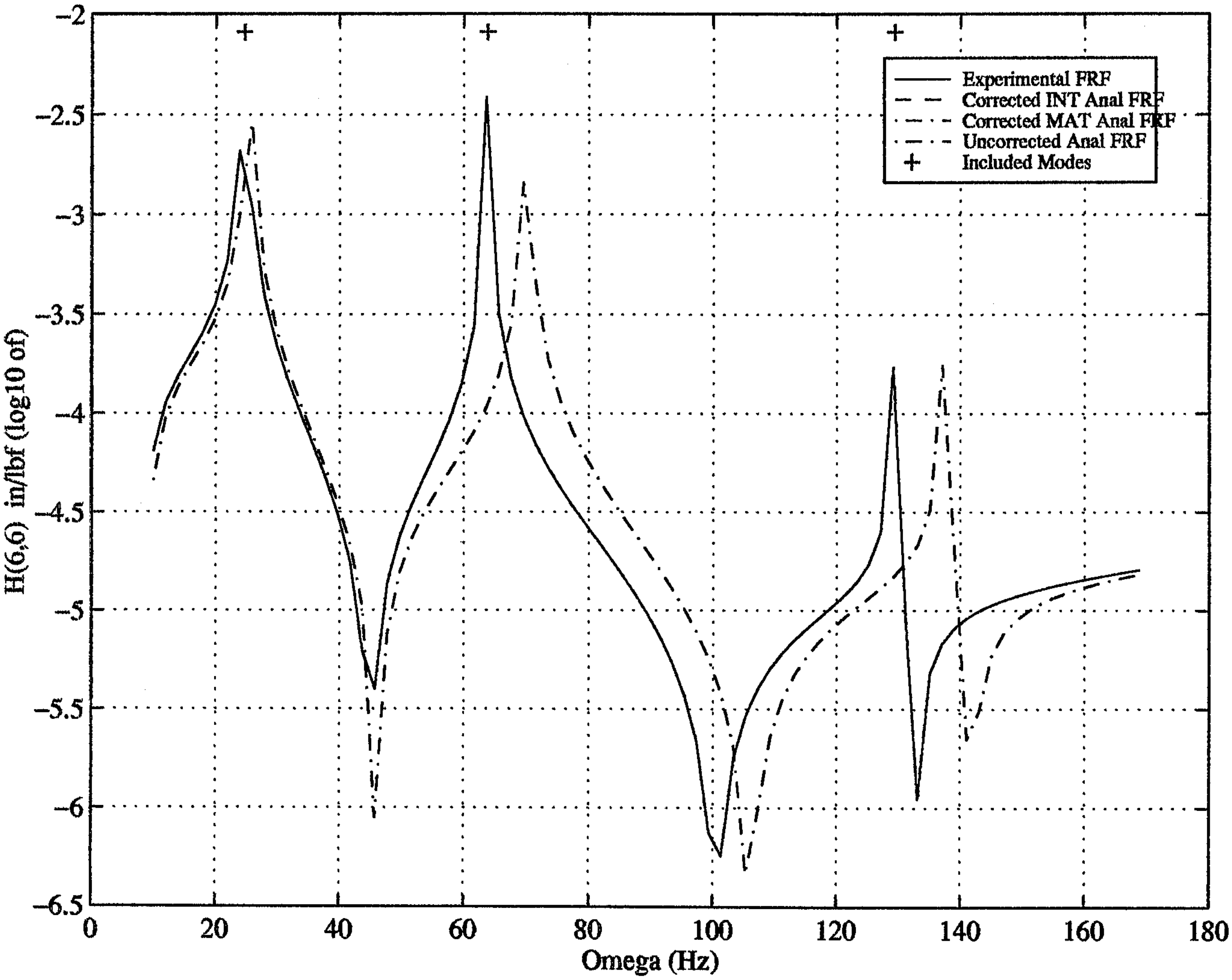

Figure 6- 11 Experimental FRF vs multiple mode matrix and integral solution computed at modes 1 through 3 using a 1 Hz bandwidth with a single point frequency samplings at each mode for a spatially complete beam.

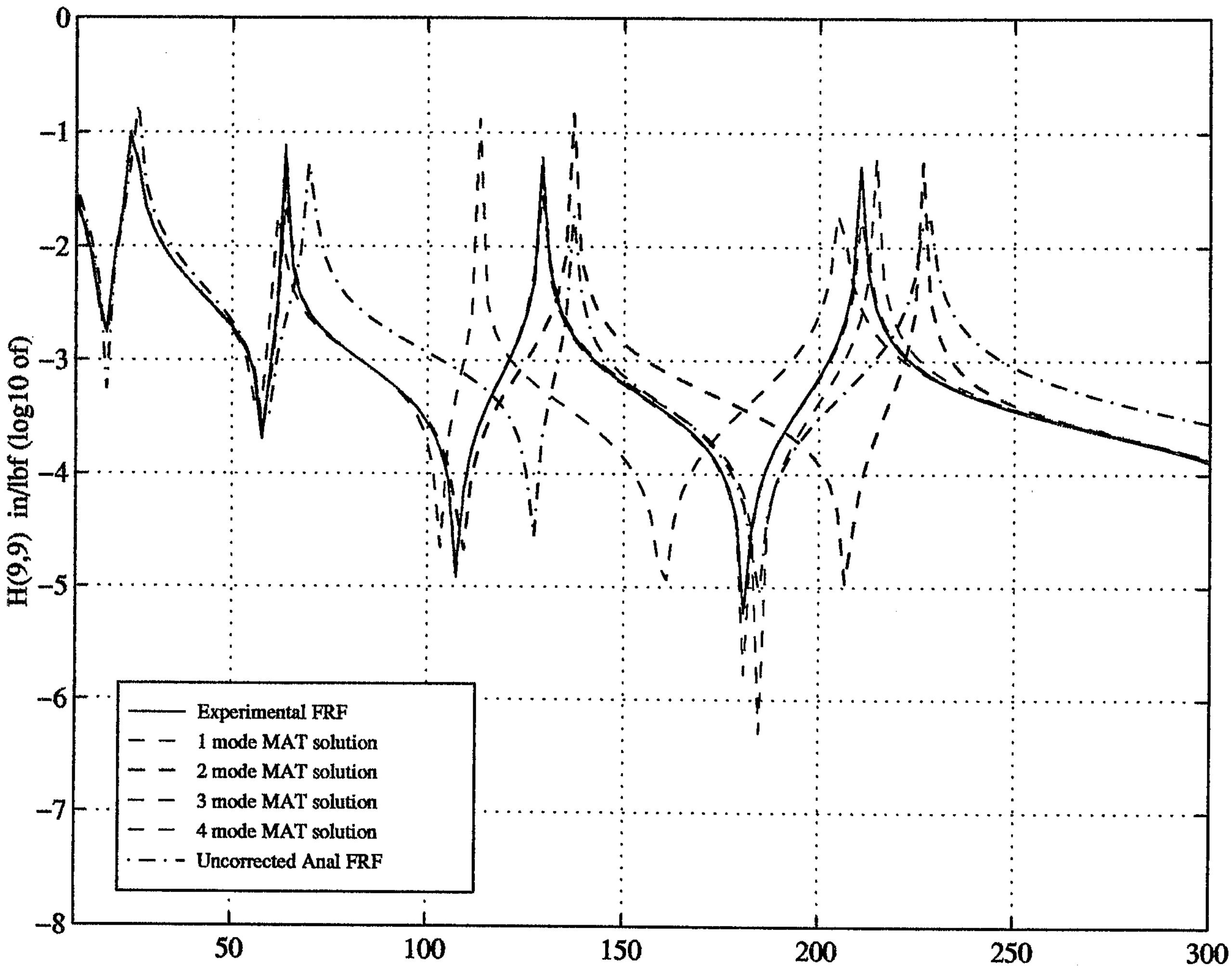

Figure 6- 12 Experimental FRF vs multiple mode matrix solutions computed over 1, 2, 3, and 4 modes using a 1 Hz bandwidth with single point frequency samplings at each mode for a spatially complete beam.

VII. CONCLUSIONS / RECOMMENDATIONS

A. SUMMARY

Frequency Domain Structural Identification using the structural synthesis transformation was performed on a simulated experimental free-free beam model. The identification was performed for both the spatially complete and the spatially incomplete case.

- SST based structural identification provides an exact solution for the identification of FE modeling errors given spatially complete data.

- For spatially incomplete systems SST based structural identification provides a frequency dependent solution which is not suitable for finite element modeling error correcting.

- Using both matrix and integral based techniques the SST can provide single mode solutions which are frequency independent correction matrices ΔK, ΔM, ΔC which approximately corrects the reduced FE model within a frequency bandwidth of the experimental system mode under consideration.

B. CONCLUSIONS

This thesis has clearly demonstrated that for spatially incomplete systems, single mode frequency independent solutions can be found which correct the reduced finite element model in a neighborhood of a given mode of the experimental system. It has also been shown that the concept of a single mode solution can be expanded to that of multiple mode solutions which are frequency independent correction matrices which approximately corrects the reduced FE model throughout a frequency bandwidth which includes more than 1 mode of the experimental system.

C. RECOMMENDATION

The purpose of this thesis was to find frequency independent multiple mode solutions which could be used to approximately correct reduced finite element models.

Although satisfactory results were obtained, investigation is still required in the following areas:

- Determine the relationship between errors in a full FE model and those of an associated reduced FE model.

- Determine a method to 'pullback' reduced FE model correction to full order FE model corrections.

- Investigate use of the integral formulation of reference (5) in analysis of the O-set system.

LIST OF REFERENCES

1. Gordis, J.H., Bielwa, R.L., and Flannery, W.G., " A General Theory for Frequency Domain Structural Synthesis", Journal of Sound and Vibration, 150(1), Sep 1989, pp.139-158.

2. Guyan, R.J., "Reduction of Stiffness and Mass Matrices", Journal of the American Institute of Aeronautics and Astronautics, Vol. 3, Feb 1965, p. 380.

3. Irons, B., "Structural Eigenvalue Problems: Elimination of Unwanted Variables", Journal of the American Institute of Aeronautics and Astronautics, Vol. 3, May 1965.

4. Gordis, J.H., "Spatial, Frequency Domain Updating of Linear, Structural Dynamic Models", AIAA-93-1652-CP, 1993, pp. 3050-3058.

5. Hagood, W.H., Crawley, E.F., "Approximate Frequency Domain Analysis for Linear Damped Space Structures", Journal of the American Institute of Aeronautics and Astronautics, Vol. 28, Nov 1990, pp. 1953-1961.

6. Bellman, R., Introduction to Matrix Analysis, McGraw-Hill, New York, 1960.

APPENDIX

The following is a brief description of MATLAB routines employed in this thesis:

•SST.M - Generates MAT files containing spatially complete or spatially incomplete experimental FRF, analytic FRF, localization matrix, and SST solutions.

•PLOTSST.M - Uses files produced by SST.M to generate plots used in chapters 3 and 4.

•CHAP3.M - Sets parameters and calls SST.M and PLOTSST.M to generate figures for chapter 3.

•CHAP4.M - Sets parameters and calls SST.M and PLOTSST.M to generate figures for chapter 4.

•CHAP5.M - Generate figures for chapter 5.

•CHAP6.M - Generate figures for chapter 6.

•SETUP.M - Generates MAT files for FE models.

•BEAMMDL.M - Setup FE models.

•FSTATIC.M - Perform static reduction of mass and stiffness matrices.

•FIRS_TAM.M - Perform IRS reduction of mass and stiffness matrices.

•FREQMODE.M - Returns FE model frequencies.

•NDX3D.M - Indexes a series of 2-D matrices into a single 3-D matrix.

•INTSUB.M - Decomposes impedance matrix into mass, stiffness, and damping matrices using integral techniques.

•MYTRAP.M - Computes integrals using trapezoidal method

SST.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%General setup%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;clc     ;%%clear workspace.
closeall      ;%close any open figure windows
;
j=sqrt(-1)     ;
load sstconf    ;%load A-set and O-set
aset(oset)=[];
aset_size=length(aset);
oset_size=length(oset);
load beamdata
if length(oset)== 0
  complete=1;
else
  complete=0;
end

%%setup plot labels%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (oset==[])&(posofMasserr(1)~=0)&(posofStifferr(1)~=0)
  casename=('Spatially Complete Mass & Stiffness error');
elseif (oset==[])&(posofMasserr(1)~=0)
  casename=('Spatially Complete Mass error');
elseif (oset==[])&(posofStifferr(1)~=0)
  casename=('Spatially Complete Stiffness error');
elseif (posofMasserr(1)~=0)&(posofStifferr(1)~=0)
  casename=('Spatially InComplete Mass & Stiffness error');
elseif (posofMasserr(1)~=0)
  casename=('Spatially InComplete Mass error');
elseif (posofStifferr(1)~=0)
  casename=('Spatially InComplete Stiffness error');
end

%%get reduced A set & O set frequencies%%%%%%%%%%%%%

kexto=k_anal(oset,oset);  %%  stiffness
mexto=m_anal(oset,oset);  %%  and mass matrices
if complete
  kstat=k_anal;
  mstat=m_anal;
else
  if static == 0
          [kstat,mstat]=fstatic(k_anal,m_anal,oset,aset);%% get reduced K & M
  elseif static == 1
          [kstat,mstat]=firs_tam(k_anal,m_anal,oset,aset);%% get reduced K & M
  else
          kstat=k_anal(aset,aset);
          mstat=m_anal(aset,aset);
  end
end
cstat=sqrt(-1)*struc_damping.*kstat;
[u,lambdaa,c]=freqmode(k_anal,m_anal)   ;%get freqs
[u,lambdared,c]=freqmode(kstat,mstat) ;%
[u,lambdax,c]=freqmode(k_exp,m_exp)     ;%
[u,lambdaexto,c]=freqmode(kexto,mexto);%
omegaa=sqrt(lambdaa);
omegax=sqrt(lambdax);
omegared=sqrt(lambdared);
omegaexto=sqrt(lambdaexto);
for i=1:4    % this to delete fixed body modes
```

```
    if real(omegax(1)) < 10^(-3)
            omegax=omegax(2:length(omegax));
    end
    if real(omegaa(1)) < 10^(-3)
      omegaa=omegaa(2:length(omegaa));
    end
    if complete==0
            if real(omegared(1)) < 10^(-3)
              omegared=omegared(2:length(omegared));
            end
            if real(omegaexto(1)) < 10^(-3)
              omegaexto=omegaexto(2:length(omegaexto));
            end
  end
end

%%%%%find true mass and stiffness errors%%%%%%%%%%%%%%%%%%%
true_stiffness=k_exp-k_anal;
stiffness_cset=find(diag(true_stiffness));
true_mass=m_exp-m_anal;
mass_cset=find(diag(true_mass));
true_damping=c_exp-c_anal;
damping_cset=find(diag(true_damping));
if length(mass_cset) > 0
  for i=1:length(mass_cset)
          x=find(stiffness_cset==mass_cset(i));
          if ~(length(x) > 0)
            stiffness_cset=[stiffness_cset; mass_cset(i)];
          end
  end
end
if length(damping_cset) > 0
  for i=1:length(damping_cset)
          x=find(stiffness_cset==damping_cset(i));
          if ~(length(x) > 0)
            stiffness_cset=[stiffness_cset; damping_cset(i)];
          end
  end
end
true_cset=sort(stiffness_cset)
save true_errs true_cset true_mass true_stiffness true_damping
clear true_mass true_stiffness true_damping
clear area eeii force g gc gcx gk gkx gm gmx goblc goblcx goblk goblkx
clear goblm goblmx ke lambdaa lambdaexto lambdared lambdax lumpdamp
clear lumpmass lumpspring me pho posofDamperr posofMasserr posofStifferr
clear stiffness_cset

%%%setup frequency range for analysis%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
freqtop=omegax(highmode)+.1*omegax(highmode);
freqbottom=omegax(lowmode)-.1*omegax(lowmode);
freqbottom=10*2*pi;   %start at 10 hz
freqtop=1200*2*pi;    %end at 1200 hz
%%number of points to plot

numpoints=400;%%%%%SET NUMBER OF POINTS TO USE FOR THE SIMULATION
fineness=numpoints;

w=freqbottom:(freqtop-freqbottom)/(numpoints-1):freqtop;

%%Locate C set%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
keepgoing='n'                ;
wfreq=omegax(lowmode) ;
nextmode=lowmode;
tol=1                        ;
defaultstepsize=.1;
while (keepgoing=='n')
  z_anal_red=kstat+j*wfreq*cstat-wfreq^2*mstat ;
  h_anal_red=inv(z_anal_red)        ;
  z_exp=k_exp+j*wfreq*c_exp-wfreq^2*m_exp  ;
  h_exp=inv(z_exp)                  ;
  h_exp=h_exp(aset,aset)              ;
  L=z_anal_red*(h_anal_red-h_exp)*z_anal_red                    ;
  ci=find(abs(diag(L))>tol)        ;
  temp_cset_size=length(ci);
  if length(ci) >= 1
          cset_size=length(ci)              ;
          figure(1)
          plot(aset, abs(diag(L)))        ;
          hold on
          plot(aset(ci),zeros(1,length(ci)),'x')
          tallest=max(abs(diag(L)))     ;
          title(['TOL = ',num2str(tol),' Cset size = ',int2str(cset_size),' Omega = ',num2str(wfreq/(2*pi)),' Hz'])
          ylabel('lbf/in')
          if complete
            xlabel('DOF')
          else
            xlabel('ASET DOF')
          end
          hold off
          k=menu(  ' Choose an action  ',...
                'Increase Tolerance ',...
                      'Decrease Tolerance ',...
                      'Increase frequency ',...
                      'Decrease frequency ',...
                      '   Previous mode   ',...
                      '    Next Mode      ',...
                      'Set Print switch on',...
                      ' Increase stepsize ',...
                      ' Decrease stepsize ',...
                      '       Go          ');
          if k==1
            stepsize=defaultstepsize;
            ctr=0;
            while length(ci) >= temp_cset_size
                    ctr=ctr+1;
                    if ctr==10
                stepsize=stepsize*10;
                      ctr=0 ;
                    end
                    tol=min(abs(tol+stepsize*(tol)),tallest) ;
                    ci=find(abs(diag(L))>tol)        ;
                    if length(ci) >= 1
                      cset_size=length(ci)              ;
                      figure(1)
                      plot(aset, abs(diag(L)))        ;
                      hold on
                      plot(aset(ci),zeros(1,length(ci)),'x')
                      tallest=max(abs(diag(L)))     ;
                      title(['TOL = ',num2str(tol),' Cset size = ',int2str(cset_size),' Omega = ',num2str(wfreq/(2*pi)),' Hz'])
                      ylabel('lbf/in')
```

```
                if complete
                        xlabel('DOF')
                else
                        xlabel('ASET DOF')
                end
                hold off
              end
      end
    elseif k==2
      stepsize=defaultstepsize;
      ctr=0;
      while length(ci) <= temp_cset_size
              ctr=ctr+1;
              if ctr==10;
          stepsize=stepsize*2;
                ctr=0;
              end
                tol=max(abs(tol-stepsize*(tol)),.00001) ;
                ci=find(abs(diag(L))>tol)        ;
                if length(ci) >= 1
                        cset_size=length(ci)             ;
                        figure(1)
                        plot(aset, abs(diag(L)))        ;
                        hold on
                        plot(aset(ci),zeros(1,length(ci)),'x')
                        tallest=max(abs(diag(L)))     ;
                        title(['TOL = ',num2str(tol),' Cset size = ',int2str(cset_size),' Omega = ',num2str(wfreq/(2*pi)),'
Hz'])
                        ylabel('lbf/in')
                        if complete
                    xlabel('DOF')
                        else
                          xlabel('ASET DOF')
                        end
                        hold off
                end
              end
    elseif k==3
      while length(ci) == temp_cset_size
              wfreq=min(wfreq+.00001*(freqtop-wfreq),freqtop);
              z_anal_red=kstat+j*w(count)*cstat-wfreq^2*mstat ;
              h_anal_red=inv(z_anal_red)        ;
              z_exp=k_exp+j*wfreq*c_exp-wfreq^2*m_exp   ;
              h_exp=inv(z_exp)                  ;
              h_exp=h_exp(aset,aset)             ;
              L=z_anal_red*(h_anal_red-h_exp)*z_anal_red                    ;
              tallest=max(abs(diag(L)))     ;
              tol=min(tol+.001*tallest,tallest) ;
              ci=find(abs(diag(L))>tol)       ;
        cset_size=length(ci)             ;
        figure(1)
        plot(aset, abs(diag(L)))        ;
    hold on
        plot(aset(ci),zeros(1,length(ci)),'x')
        title(['TOL = ',num2str(tol),' Cset size = ',int2str(cset_size),' Omega = ',num2str(wfreq/(2*pi)),' Hz'])
        ylabel('lbf/in')
        if complete
                xlabel('DOF')
        else
                xlabel('ASET DOF')
```

```
                end
                hold off
           end
              elseif k==4
                    wfreq=max(wfreq-.05*(wfreq-freqbottom),freqbottom);
              elseif k==5
                    nextmode=max(lowmode,nextmode-1);
                    wfreq=omegax(nextmode);
              elseif k==6
                    nextmode=min(highmode,nextmode+1);
                    wfreq=omegax(nextmode);
              elseif k==7
                    pswitch='y'          ;
        elseif k==8
              defaultstepsize=defaultstepsize/10;
        elseif k==9
              defaultstepsize=defaultstepsize*10;
      else
            keepgoing='y'          ;
            cset=aset(ci);
            cset_rel=ci                    ;
      end
    else
          tol=tol*.9;
    end
  end

  close(1)
  save extrtset cset cset_rel
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%
  %%%%%%%%%%%%%%%%%%FORCE THE CSET%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %cset=[7 9 11 13]  %%%%%%%%THE INCOMPLETE CASE CSET%%%%%%%%%%%%
  %cset_rel=[4 5 6 7]%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%
  %cset=[4 5 6 7 8 9 10 11]%%%%%%%%%%%%%THE COMPLETE CASE CSET%%%
  %cset_rel=[4 5 6 7 8 9 10 11]%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %%%
  cset_size=length(cset);
  save L L
  clear L
  clear kexta kexto mexta mexto
  pack
  skyfull=zeros(numdof,numdof);
  count=1  ;
  for index1=1:numdof
    for index2=index1:numdof
          skyfull(index1,index2)=count;
          skyfull(index2,index1)=count;
          count=count+1;
    end
  end
  skyred=zeros(aset_size,aset_size);
  count=1   ;
  for index1=1:aset_size
    for index2=index1:aset_size
          skyred(index1,index2)=count;
          skyred(index2,index1)=count;
```

```
            count=count+1;
        end
    end
    skycset=zeros(cset_size,cset_size);
    count=1   ;
    for index1=1:cset_size
        for index2=index1:cset_size
            skycset(index1,index2)=count;
            skycset(index2,index1)=count;
            count=count+1;
        end
    end

    full_holder=zeros(1,(numdof*(numdof+1))/2);
    red_holder=zeros(1,(aset_size*(aset_size+1))/2);
    cset_holder=zeros(1,(cset_size*(cset_size+1))/2);

    if meters
        waitbar_handle=waitbar(0,'Computing Experimental FRF');
    else
        disp('Getting experimental FRF')
    end
    H_EXP=[];
    for count=1:numpoints
        if meters
            waitbar(count/numpoints);
        end
        z_exp=k_exp+j*w(count)*c_exp-w(count)^2*m_exp;
        h_exp=inv(z_exp);
        h_exp=h_exp(aset,aset);
        skyindex=1;
        for index1=1:aset_size
            for index2=index1:aset_size
                red_holder(skyindex)=h_exp(index1,index2);
                skyindex=skyindex+1;
            end
        end
        H_EXP=[H_EXP red_holder'];
    end
    if meters
        close(waitbar_handle)
    end
    save H_EXP H_EXP
    clear H_EXP
    pack

    if meters
        waitbar_handle=waitbar(0,'Computing Experimental Impedence');
    else
        disp('Getting experimental Impedence')
    end
    Z_EXP=[];
    for count=1:numpoints
        if meters
            waitbar(count/numpoints);
        end
        z_exp=k_exp+j*w(count)*c_exp-w(count)^2*m_exp;
        h_exp=inv(z_exp);
        h_exp=h_exp(aset,aset);
        z_exp=inv(h_exp);
```

```
      skyindex=1 ;
      for index1=1:aset_size
              for index2=index1:aset_size
                red_holder(skyindex)=z_exp(index1,index2);
                skyindex=skyindex+1;
              end
      end
      Z_EXP=[Z_EXP red_holder'];
    end
    if meters
      close(waitbar_handle)
    end
    save Z_EXP Z_EXP
    clear Z_EXP
    pack

    if meters
      waitbar_handle=waitbar(0,'Computing Reduced Analytic FRF');
    else
      disp('Getting Reduced FRF')
    end
    H_ANAL_RED=[];
    for count=1:numpoints
      if meters
              waitbar(count/numpoints);
      end
      z_anal_red=kstat+j*w(count)*cstat-w(count)^2*mstat;
      h_anal_red=inv(z_anal_red);
      skyindex=1  ;
      for index1=1:aset_size
              for index2=index1:aset_size
                red_holder(skyindex)=h_anal_red(index1,index2);
                skyindex=skyindex+1;
              end
      end
      H_ANAL_RED=[H_ANAL_RED red_holder'];
    end
    if meters
      close(waitbar_handle)
    end
    save H_ANAL_R H_ANAL_RED
    clear Z_ANAL_RED H_ANAL_RED kexta kexto mexta mexto
    pack

    %% compute and save L matrix diagonal as a function of frequency%%%%%%%%
    if meters
      waitbar_handle=waitbar(0,'Computing L Diagonals');
    else
      disp('Getting L Matrix')
    end
    L_DIAGS=[];
    for count=1:numpoints
      if meters
              waitbar(count/numpoints);
      end
      z_anal=k_anal+j*w(count)*c_anal-w(count)^2*m_anal;
      z_anal_red=kstat+j*w(count)*cstat-w(count)^2*mstat;
      h_anal_red=inv(z_anal_red);
      z_exp=k_exp+j*c_exp-w(count)^2*m_exp;
      h_exp=inv(z_exp);
```

```
    h_exp=h_exp(aset,aset);
    L=z_anal_red*(h_anal_red-h_exp)*z_anal_red;
    temp_l_diags=diag(L);
    L_DIAGS=[L_DIAGS temp_l_diags(:)];
  end
  if meters
    close(waitbar_handle)
  end
  save L_DIAGS L_DIAGS
  clear L_DIAGS z_anal temp_l_diags L
  clear z_exp z_anal_red u v  h_exp h_anal_red
  pack

  if meters
    waitbar_handle=waitbar(0,'Computing Z Analytic');
  else
    disp('Getting Z anal')
  end
  Z_ANAL=[];
  for count=1:numpoints
    if meters
            waitbar(count/numpoints);
    end
    z_anal=k_anal+j*w(count)*c_anal-w(count)^2*m_anal;
    skyindex=1   ;
    for index1=1:numdof
            for index2=index1:numdof
              full_holder(skyindex)=z_anal(index1,index2);
              skyindex=skyindex+1;
            end
    end
    Z_ANAL=[Z_ANAL full_holder'];
  end
  if meters
    close(waitbar_handle)
  end
  save Z_ANAL Z_ANAL
  clear Z_ANAL z_anal
  pack
  if meters
    waitbar_handle=waitbar(0,'Computing Reduced Z Analytic');
  else
    disp('Getting Reduced Z anal')
  end
  Z_ANAL_RED=[];
  for count=1:numpoints
    if meters
            waitbar(count/numpoints);
    end
    z_anal_red=kstat+j*w(count)*cstat-w(count)^2*mstat;
    skyindex=1    ;
    for index1=1:aset_size
            for index2=index1:aset_size
              red_holder(skyindex)=z_anal_red(index1,index2);
              skyindex=skyindex+1;
            end
    end
    Z_ANAL_RED=[Z_ANAL_RED red_holder'];
  end
  if meters
```

```
    close(waitbar_handle)
  end
  save Z_ANAL_R Z_ANAL_RED
  clear Z_ANAL_RED z_anal_red
  pack

  %%%compute DZ%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  if meters
    waitbar_handle=waitbar(0,'Computing DZ...');
  else
    disp('Getting DZ')
  end
  DZ=[];
  for count=1:numpoints
    if meters
            waitbar(count/numpoints);
    end
    z_anal_red=kstat+j*w(count)*cstat-w(count)^2*mstat;
    h_anal_red=inv(z_anal_red);
    z_exp=k_exp+j*w(count)*c_exp-w(count)^2*m_exp;
    h_exp=inv(z_exp);
    h_exp=h_exp(aset,aset);
    hacc=h_anal_red(cset_rel,cset_rel);
    hxcc=h_exp(cset_rel,cset_rel);
    dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
    skyindex=1   ;
    for index1=1:cset_size
            for index2=index1:cset_size
              cset_holder(skyindex)=dz(index1,index2);
              skyindex=skyindex+1;
            end
    end
    DZ=[DZ cset_holder'];
  end
  if meters
    close(waitbar_handle)
  end
  save DZ DZ
  clear hacc hxcc dz h_exp z_anal_red h_anal_red z_exp
  pack

  %%%decompose DZ into DM, DK, & DC%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  if meters
    waitbar_handle=waitbar(0,'Decomposing DZ...');
  else
    disp('Decomposing DZ')
  end
  DK=[];
  DM=[];
  DC=[];
  dz1=zeros(cset_size,cset_size);
  dz2=zeros(cset_size,cset_size);
  dz3=zeros(cset_size,cset_size);
  for i=1:numpoints-2
    if meters
      waitbar(i/(numpoints-2))
    end
    dztemp=DZ(ndx3d([1 cset_size*(cset_size+1)/2 numpoints],1,1:cset_size*(cset_size+1)/2,i));
    skyindex=1;
    for index1=1:cset_size
```

```
          for index2=index1:cset_size
            dz1(index1,index2)=dztemp(skyindex);
            dz1(index2,index1)=dztemp(skyindex);
            skyindex=skyindex+1;
          end
    end
    dztemp=DZ(ndx3d([1 cset_size*(cset_size+1)/2 numpoints],1,1:cset_size*(cset_size+1)/2,i+1));
    skyindex=1;
    for index1=1:cset_size
          for index2=index1:cset_size
            dz2(index1,index2)=dztemp(skyindex);
            dz2(index2,index1)=dztemp(skyindex);
            skyindex=skyindex+1;
          end
    end
    dztemp=DZ(ndx3d([1 cset_size*(cset_size+1)/2 numpoints],1,1:cset_size*(cset_size+1)/2,i+2));
    skyindex=1;
    for index1=1:cset_size
          for index2=index1:cset_size
            dz3(index1,index2)=dztemp(skyindex);
            dz3(index2,index1)=dztemp(skyindex);
            skyindex=skyindex+1;
          end
    end
    temp=[eye(cset_size) -w(i)^2*eye(cset_size)  +j*w(i)*eye(cset_size);
    eye(cset_size) -w(i+1)^2*eye(cset_size) +j*w(i+1)*eye(cset_size);
    eye(cset_size) -w(i+2)^2*eye(cset_size) +j*w(i+2)*eye(cset_size)]\[dz1;dz2;dz3];
    ktemp=temp(1:cset_size,:);
    mtemp=temp(cset_size+1:2*cset_size,:);
    ctemp=temp(2*cset_size+1:3*cset_size,:);

    skyindex=1    ;
    for index1=1:cset_size
          for index2=index1:cset_size
            cset_holder(skyindex)=ktemp(index1,index2);
            skyindex=skyindex+1;
          end
    end
    DK=[DK cset_holder(:)];

    skyindex=1    ;
    for index1=1:cset_size
          for index2=index1:cset_size
            cset_holder(skyindex)=mtemp(index1,index2);
            skyindex=skyindex+1;
          end
    end
    DM=[DM cset_holder(:)];

    skyindex=1    ;
    for index1=1:cset_size
          for index2=index1:cset_size
            cset_holder(skyindex)=ctemp(index1,index2);
            skyindex=skyindex+1;
          end
    end
    DC=[DC cset_holder(:)];
end
if meters
  close(waitbar_handle)
```

```
end
save DM DM
save DK DK
save DC DC
clear DC DK DM
save exp c_exp k_exp m_exp
clear c_exp k_exp m_exp
save stat kstat mstat cstat
clear kstat mstat cstat
save anal k_anal m_anal c_anal
clear k_anal m_anal c_anal
pack

if complete
  load Z_ANAL
  Z=Z_ANAL;
  clear Z_ANAL
else
  load Z_ANAL_R
  Z=Z_ANAL_RED;
  clear Z_ANAL_RED
end
CORRHA=[];
tempza=zeros(aset_size,aset_size);
tempz=zeros(cset_size,cset_size);
if meters
  waitbar_handle=waitbar(0,'Installing DZ...');
else
  disp('Installing DZ')
end
for i=1:numpoints
  if meters
        waitbar(i/numpoints)
  end
  ztemp=Z(ndx3d([1 aset_size*(aset_size+1)/2 (numpoints-(i-1))],1,1:aset_size*(aset_size+1)/2,1));
  skyindex=1;
  for index1=1:aset_size
        for index2=index1:aset_size
          tempza(index1,index2)=ztemp(skyindex);
          tempza(index2,index1)=ztemp(skyindex);
          skyindex=skyindex+1;
        end
  end
  Z(:,1)=[];
  ztemp=DZ(ndx3d([1 cset_size*(cset_size+1)/2 (numpoints-(i-1))],1,1:cset_size*(cset_size+1)/2,1));
  skyindex=1;
  for index1=1:cset_size
        for index2=index1:cset_size
          tempz(index1,index2)=ztemp(skyindex);
          tempz(index2,index1)=ztemp(skyindex);
          skyindex=skyindex+1;
        end
  end
  tempza(cset_rel,cset_rel)=tempza(cset_rel,cset_rel)+tempz;
  DZ(:,1)=[];
  tempha=inv(tempza);
  CORRHA=[CORRHA tempha(:)];
end
if meters
  close(waitbar_handle)
```

```
  end
  clear DZ Z
  save CORRHA CORRHA
  clear CORRHA

  load DK;
  load DM;
  load DC;

 load stat
 CORRHAD=[];
 tempza=zeros(aset_size,aset_size);
 tempz=zeros(cset_size,cset_size);
 tempk=zeros(cset_size,cset_size);
 tempm=zeros(cset_size,cset_size);
 tempc=zeros(cset_size,cset_size);
 if meters
    waitbar_handle=waitbar(0,'Installing DK, DM, & DC...');
 else
   disp('Installing DK, DM, DC')
 end
 for i=1:numpoints-2
   if meters
           waitbar(i/(numpoints-2))
   end
   kcorrected=kstat;
   mcorrected=mstat;
   ccorrected=cstat;

   temp=DK(ndx3d([1 cset_size*(cset_size+1)/2 (numpoints-(2))],1,1:cset_size*(cset_size+1)/2,1));
   skyindex=1;
   for index1=1:cset_size
           for index2=index1:cset_size
             tempk(index1,index2)=temp(skyindex);
             tempk(index2,index1)=temp(skyindex);
             skyindex=skyindex+1;
           end
   end

   DK(:,1)=[];
   temp=DM(ndx3d([1 cset_size*(cset_size+1)/2 (numpoints-(2))],1,1:cset_size*(cset_size+1)/2,1));
   skyindex=1;
   for index1=1:cset_size
           for index2=index1:cset_size
             tempm(index1,index2)=temp(skyindex);
             tempm(index2,index1)=temp(skyindex);
             skyindex=skyindex+1;
           end
   end

   DM(:,1)=[];
   temp=DC(ndx3d([1 cset_size*(cset_size+1)/2 (numpoints-(2))],1,1:cset_size*(cset_size+1)/2,1));
   skyindex=1;
   for index1=1:cset_size
           for index2=index1:cset_size
             tempc(index1,index2)=temp(skyindex);
             tempc(index2,index1)=temp(skyindex);
             skyindex=skyindex+1;
           end
   end
```

```
    DC(:,1)=[];
    kcorrected(cset_rel,cset_rel)=kcorrected(cset_rel,cset_rel)+tempk;
    mcorrected(cset_rel,cset_rel)=mcorrected(cset_rel,cset_rel)+tempm;
    ccorrected(cset_rel,cset_rel)=ccorrected(cset_rel,cset_rel)+tempc;

    tempza=kcorrected+j*w(i+1)*ccorrected-w(i+1)^2*mcorrected;
    tempha=inv(tempza);
    CORRHAD=[CORRHAD tempha(:)];
  end
  if meters
    close(waitbar_handle)
  end
  clear tempk tempm tempc tempza tempha
  clear kcorrected mcorrected ccorrected
  save CORRHAD CORRHAD
  clear CORRHAD DK DM DC

  %%%save the
  Workspace%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  save INT
```

PLOTSST.M

```
clear
closeall
load INT
titles=0;
if pswitch=='y'
  whitebg('white')
  close
end
h=0;
if complete
  fignum=1;
else
  fignum=21;
end
%% write frequencies and system diag to diary file prt.out%%%%%%%%%%%%%%%%
if exist('fig000.out')
  delete fig000.out
end
diary fig000.out
fprintf('     \n')
fprintf('     \n')
fprintf('A set\n')
for index=1:length(aset)
  fprintf([blanks(4-length(int2str(aset(index)))),int2str(aset(index))])
  if rem(index,12)==0
    fprintf('\n')
  end
end
fprintf('\n')
fprintf('\n')
fprintf('O set\n')
for index=1:length(oset)
  fprintf([blanks(4-length(int2str(oset(index)))),int2str(oset(index))])
  if rem(index,12)==0
    fprintf('\n')
  end
end
fprintf('\n')
fprintf('\n')
fprintf('Computed C set\n')
for index=1:length(cset)
  fprintf([blanks(4-length(int2str(cset(index)))),int2str(cset(index))])
  if rem(index,12)==0
    fprintf('\n')
  end
end
fprintf('\n')
fprintf('\n')
fprintf('True C set\n')
for index=1:length(true_cset)
  fprintf([blanks(4-length(int2str(true_cset(index)))),int2str(true_cset(index))])
  if rem(index,12)==0
    fprintf('\n')
  end
end
fprintf('\n')
fprintf('\n')
bigger=max(length(omegaa),length(omegax));
if complete
  allfreqs=[ ...
```

```
    [omegaa/(2*pi); zeros(bigger-length(omegaa),1)] ...
    [omegax/(2*pi); zeros(bigger-length(omegax),1)] ...
        ];
  else
    allfreqs=[ ...
    [omegaa/(2*pi); zeros(bigger-length(omegaa),1)     ]...
    [omegax/(2*pi); zeros(bigger-length(omegax),1)      ]...
    [omegared/(2*pi); zeros(bigger-length(omegared),1)  ] ...
    [omegaexto/(2*pi); zeros(bigger-length(omegaexto),1)]...
        ];
  end
  [aaa bbb]=size(allfreqs);
  fprintf('System Frequencies (Hz)\n')
  if complete
    fprintf('     Anal    Exp\n')
    bbb=2;
  else
    fprintf('  Anal    Exp    Red    Oset\n')
    bbb=4;
  end
  for index1=1:aaa
    prtline=[];
    for index2=1:bbb
      if allfreqs(index1,index2)==0
        prtfreq='        ';
      else
        prtfreq=[blanks(8-length(sprintf('%4g',allfreqs(index1,index2)))),...
        sprintf('%4g',allfreqs(index1,index2))];
      end
      prtline=[prtline, prtfreq];
    end
    prtline=[prtline, '\n'];
    fprintf(prtline)
  end
  diary off
  %dos('type fig000.out > lpt1 &');
  xx=omegax(find(omegaa(lowmode) <= omegax & omegax <= omegaa(highmode)))/(2*pi);
  yx1=ones(length(xx),1);
  aa=omegaa(find(omegaa(lowmode) <= omegaa & omegaa <= omegaa(highmode)))/(2*pi);
  ya1=ones(length(aa),1);
  red=omegared(find(omegaa(lowmode) <= omegared & omegared <= omegaa(highmode)))/(2*pi);
  yred1=ones(length(red),1);
  exto=omegaexto(find(omegaa(lowmode) <= omegaexto & omegaexto <= omegaa(highmode)))/(2*pi);
  yexto1=ones(length(exto),1);
  if complete
    x0=max([xx(length(xx)) aa(length(aa))]);
  else
    x0=max([xx(length(xx)) aa(length(aa)) red(length(red)) exto(length(exto))]);
  end
  clear omegax omegaa omegared omegaexto
  clear L Z_ANAL c_anal c_exp conn h_exp h_anal_red k_anal k_exp
  clear kexta kstat kexto m_anal m_exp mexto mstat temp_l_diags
  if complete
    mid_index=round(length(cset)/2);
  else
    mid_index=round(length(cset)/2);
  end
  load H_ANAL_R
  load H_EXP
```

```
h=figure(h+1);
fignum=fignum+1 ;
plot(w/(2*pi),log10(abs(H_ANAL_RED(ndx3d([1 aset_size*(aset_size+1)/2 numpoints],...
   1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r--',...
   w/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 numpoints],...
   1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'b-.')
v=axis;
ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (Log10 of)'])
xlabel('Omega  (Hz)')
if titles
  title('Analytic FRF vs Experimental FRF')
end
hold on
if abs(v(4)-v(3)) <= 1
  v2(1)=v(1);
  v2(2)=v(2);
  v2(3)=v(3)-.1*abs(v(4));
  v2(4)=v(4)+.1*abs(v(4));
  axis(v2);
end
grid on
hh=slegend('ANALYTIC','EXPERIMENTAL');
axes(hh);
hold off
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end

clear H_ANAL_RED
clear H_EXP

if complete
  h=figure(h+1);
  fignum=fignum+1;
else
  h=figure(h+2);
  fignum=fignum+2;
end
load L
plot(aset,abs(diag(L)),aset,abs(diag(L)),'*')
ylabel(' L(DOF) lbf/in')
if complete
  xlabel('DOF')
else
  xlabel('ASET DOF')
end
grid on
if titles
  title(['Localization Matrix Diagonal at Omega = ',num2str(wfreq/(2*pi)), ' Hz']);
end
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end
clear L

h=figure(h+1);
fignum=fignum+1;
load L_DIAGS
```

```
mesh(w/(2*pi),aset,log10(abs(L_DIAGS)))
if titles
  title('Frequency Dependence of Localization Matrix Diagonals')
end
xlabel('Omega  (Hz)')
ylabel('DOF')
zlabel('L(DOF, omega) lbf/in (Log10 of)')
grid on
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end

h=figure(h+1);
fignum=fignum+1;
subplot(2,1,1)
plot(w/(2*pi),log10(L_DIAGS(ndx3d([aset_size 1 numpoints],cset_rel(mid_index+2),1,':'))))
ylabel(['Error Coord L(',int2str(cset(mid_index+2)),',',int2str(cset(mid_index+2)),')'])
if titles
  title('lbf/in (Log10 of)' );
end
grid on
v1=axis;
subplot(2,1,2)

if length(cset_rel) < length(aset)
  holdset=1:length(aset);
  holdset(cset_rel)=[];
  index=round(length(holdset)/2);

  plot(w/(2*pi),log10(L_DIAGS(ndx3d([aset_size 1 numpoints],holdset(index),1,':'))))
  ylabel(['Non-Error Coord L(',int2str(aset(holdset(index))),',',int2str(aset(holdset(index))),')'])
  if titles
    title('Frequency Dependence of L Matrix Non-Error set Diagonal Elements' )
  end
  v=axis;
  hold on
  v2(1)=v(1);
  v2(2)=v(2);
  v2(3)=v(3)+((v(4)-v(3))*.5)-(v1(4)-v1(3))*.5 ;
  v2(4)=v(3)+((v(4)-v(3))*.5)+(v1(4)-v1(3))*.5;
  axis(v2);
  grid on
end
xlabel('Omega  (Hz)')
hold off
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end
clear L_DIAGS

load Z_ANAL_R
load Z_EXP
h=figure(h+1);
fignum=fignum+1;
plot(w/(2*pi),log10(Z_ANAL_RED(ndx3d([1 aset_size*(aset_size+1)/2
numpoints],1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':'))),'r--',...
  w/(2*pi),log10(Z_EXP(ndx3d([1 aset_size*(aset_size+1)/2
numpoints],1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':'))),'b-.')
```

```
xlabel('Omega^2 (Hz)')
ylabel(['Z(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') lbf/in (Log10 of)'])
if titles
  title('Analytic Impedance vs Experimental Impedance')
end
v=axis;
hold on
if abs(v(4)-v(3)) <= 1
  v2(1)=v(1);
  v2(2)=v(2);
  v2(3)=v(3)-.1*abs(v(4));
  v2(4)=v(4)+.1*abs(v(4));
  axis(v2);
end
grid on
if complete== 0
  hh=slegend('ANALYTIC IMPEDANCE','EXPERIMENTAL IMPEDANCE');
else
  hh=slegend('ANALYTIC IMPEDANCE','EXPERIMENTAL IMPEDANCE');
end
axes(hh)
hold off
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end
clear Z_ANAL_RED Z_EXP

load true_errs
load DK
h=figure(h+1);
fignum=fignum+1;
if complete
  plot(w(1:numpoints-2)/(2*pi),DK(ndx3d([1 cset_size*(cset_size+1)/2 numpoints-2],1,skycset(mid_index,mid_index),':')),'r--
',...
    w(1:numpoints-2)/(2*pi),true_stiffness(cset(mid_index),cset(mid_index)).*ones(1,numpoints-2),'b-.')
  ylabel(['DK(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') lbf/in'])
  xlabel('Omega (Hz)')
else
  plot(w(1:numpoints-2)/(2*pi),log10(abs(DK(ndx3d([1 cset_size*(cset_size+1)/2 numpoints-
2],1,skycset(mid_index,mid_index),':')))),'r--',...
    w(1:numpoints-2)/(2*pi),log10((true_stiffness(cset(mid_index),cset(mid_index)) == 0)+...
    true_stiffness(cset(mid_index),cset(mid_index))).*ones(1,numpoints-2),'b-.')
  ylabel(['DK(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') lbf/in (Log10 of)'])
  xlabel('Omega (Hz)')
end
v=axis;
if titles
  title('Computed Stiffness Error vs True Stiffness Error')
end
v2(1)=v(1);
v2(2)=v(2);
if complete
  if abs(v(4)-v(3)) < 1
    v2(3)=v(3)-100*abs(v(4)-v(3));

    v2(4)=v(4)+100*abs(v(4)-v(3));
    v(3)=v2(3);
    v(4)=v2(4);
  end
```

```
    if abs(true_stiffness(cset(mid_index),cset(mid_index)) - v(3)) < .25*abs(v(4)-v(3))
      v2(3)=v(3)-.5*abs(v(4)-v(3)) ;
      v2(4)=v(4);
    elseif abs(true_stiffness(cset(mid_index),cset(mid_index)) - v(4)) < .25*abs(v(4)-v(3))
      v2(4)=v(4)+.5*abs(v(4)-v(3)) ;
      v2(3)=v(3);
    else
      v2(3)=v(3);
      v2(4)=v(4);
    end
  else
    if abs(log10((true_stiffness(cset(mid_index),cset(mid_index)) == 0)+...
      true_stiffness(cset(mid_index),cset(mid_index))) - v(3)) < .25*abs(v(4)-v(3))
      v2(3)=v(3)-.5*abs(v(4)-v(3)) ;
      v2(4)=v(4);
    elseif abs(log10((true_stiffness(cset(mid_index),cset(mid_index)) == 0)+...
      true_stiffness(cset(mid_index),cset(mid_index))) - v(4)) < .25*abs(v(4)-v(3))
      v2(4)=v(4)+.5*abs(v(4)-v(3)) ;
      v2(3)=v(3);
    else
      v2(3)=v(3);
      v2(4)=v(4);
    end
  end
  axis(v2)
  grid on
  hh=slegend('COMPUTED STIFFNESS ERROR','TRUE STIFFNESS ERROR');
  axes(hh)                                   ;
  if pswitch=='y'
    prtfig(fignum)
    delete(h)
  end
  clear DK
  load DM
  h=figure(h+1) ;
  fignum=fignum+1 ;
  if complete
    plot(w(1:numpoints-2)/(2*pi),DM(ndx3d([1 cset_size*(cset_size+1)/2 numpoints-2],1,skycset(mid_index,mid_index),':')),'r--
  ',...
      w(1:numpoints-2)/(2*pi),true_mass(cset(mid_index),cset(mid_index)).*ones(1,numpoints-2),'b-.')
    ylabel(['DM(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') lbf-sec^2/in'])
    xlabel('Omega (Hz)')
  else
    plot(w(1:numpoints-2)/(2*pi),log10(abs(DM(ndx3d([1 cset_size*(cset_size+1)/2 numpoints-
  2],1,skycset(mid_index,mid_index),':')))),'r--',...
      w(1:numpoints-2)/(2*pi),log10(...
      (...
      true_mass(cset(mid_index),cset(mid_index)) == 0 ...
        )+true_mass(cset(mid_index),cset(mid_index))...
        ).*ones(1,numpoints-2),'b-.')
        ylabel(['DM(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') lbf-sec^2/in (Log10 of)'])
        xlabel('Omega (Hz)')
  end
  v=axis;
  if titles
    title('Computed Mass Error vs True Mass Error')
  end
  hold on
  v2(1)=v(1);
  v2(2)=v(2);
```

```
if complete
  if abs(v(4)-v(3)) < 1
    v2(3)=v(3)-10000*abs(v(4)-v(3));
    v2(4)=v(4)+10000*abs(v(4)-v(3));
    v(3)=v2(3);
    v(4)=v2(4);
  end
  if abs(true_mass(cset(mid_index),cset(mid_index)) - v(3)) < .25*abs(v(4)-v(3))
    v2(3)=v(3)-.5*abs(v(4)-v(3));
    v2(4)=v(4);
  elseif abs(true_mass(cset(mid_index),cset(mid_index)) - v(4)) < .25*abs(v(4)-v(3))
    v2(4)=v(4)+.5*abs(v(4)-v(3));
    v2(3)=v(3);
  else
    v2(3)=v(3);
    v2(4)=v(4);
  end
else
  if abs(log10((true_mass(cset(mid_index),cset(mid_index)) == 0)+...
    true_mass(cset(mid_index),cset(mid_index))) - v(3)) < .25*abs(v(4)-v(3))
    v2(3)=v(3)-.5*abs(v(4)-v(3)) ;
    v2(4)=v(4);
  elseif abs(log10((true_mass(cset(mid_index),cset(mid_index)) == 0)+...
    true_mass(cset(mid_index),cset(mid_index))) - v(4)) < .25*abs(v(4)-v(3))
    v2(4)=v(4)+.5*abs(v(4)-v(3)) ;
    v2(3)=v(3);
  else
    v2(4)=v(4);
    v2(3)=v(3);
  end
end
axis(v2)
grid on
hh=slegend('COMPUTED MASS ERROR','TRUE MASS ERROR');
axes(hh)
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end
clear DM

load DC
h=figure(h+1);
fignum=fignum+1;
if complete
  plot(w(1:numpoints-2)/(2*pi),imag(DC(ndx3d([1 cset_size*(cset_size+1)/2 numpoints-
2],1,skycset(mid_index,mid_index),':'))),'r--',...
    w(1:numpoints-2)/(2*pi),imag(true_damping(cset(mid_index),cset(mid_index)).*ones(1,numpoints-2)),'b-.')
  ylabel(['DC(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') lbf-sec/in'])
  xlabel('Omega (Hz)')
else
  plot(w(1:numpoints-2)/(2*pi),log10(abs(DC(ndx3d([1 cset_size*(cset_size+1)/2 numpoints-
2],1,skycset(mid_index,mid_index),':')))),'r--',...
    w(1:numpoints-2)/(2*pi),log10(abs(...
    (...
    true_damping(cset(mid_index),cset(mid_index)) == 0 ...
      )+true_damping(cset(mid_index),cset(mid_index))...
       ).*ones(1,numpoints-2)),'b-.')
       ylabel(['DC(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') lbf-sec/in (Log10 of)'])
       xlabel('Omega (Hz)')
```

```
    end
    v=axis;
    if titles
      title('Computed Damping Error vs True Damping Error')
    end
    v2(1)=v(1);
    v2(2)=v(2);
    if complete
      if abs(v(4)-v(3)) < 1
        v2(3)=v(3)-100*abs(v(4)-v(3));
        v2(4)=v(4)+100*abs(v(4)-v(3));
        v(3)=v2(3);
        v(4)=v2(4);
      end
      if abs(true_damping(cset(mid_index),cset(mid_index)) - v(3)) < .25*abs(v(4)-v(3))
        v2(3)=v(3)-.25*abs(v(4)-v(3)) ;
        v2(4)=v(4);
      elseif abs(true_damping(cset(mid_index),cset(mid_index)) - v(4)) < .25*abs(v(4)-v(3))
        v2(4)=v(4)+.25*abs(v(4)-v(3)) ;
        v2(3)=v(3);
      else
        v2(3)=v(3);
        v2(4)=v(4);
      end
    else
      if abs(log10((true_damping(cset(mid_index),cset(mid_index)) == 0)+...
        true_damping(cset(mid_index),cset(mid_index))) - v(3)) < .1*abs(v(4)-v(3))
        v2(3)=(v(3)-.25*abs(v(4)-v(3))) ;
        v2(4)=v(4);
      elseif abs(log10((true_damping(cset(mid_index),cset(mid_index)) == 0)+...
        true_damping(cset(mid_index),cset(mid_index))) - v(4)) < .1*abs(v(4)-v(3))
        v2(4)=(v(4)+.25*abs(v(4)-v(3))) ;
        v2(3)=v(3);
      else
        v2(3)=v(3);
        v2(4)=v(4);
      end
    end
    axis(v2);
    grid on
    hh=slegend('COMPUTED DAMPING ERROR',TRUE DAMPING ERROR');
    axes(hh);
    if pswitch=='y'
      prtfig(fignum)
      delete(h)
    end
    clear DC true_damping true_mass true_stiffness

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%

    load H_EXP
    load CORRHA
    %for mid_index=1:min(cset_size,multplot)
    h=figure(h+1);
    fignum=fignum+1;
    plot(w/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2
    numpoints],1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r--',...
      w/(2*pi),log10(abs(CORRHA(ndx3d([aset_size aset_size numpoints],cset_rel(mid_index),cset_rel(mid_index),':')))),'b-.')
      ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (Log10 of)'])
```

```
  xlabel('Omega  (Hz)')
if titles
  title('Experimental FRF vs DZ Corrected FRF')
end
grid on
v=axis;
hh=slegend('EXPERIMENTAL FRF','CORRECTED FRF');
axes(hh)   ;
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end

clear CORRHA
load CORRHAD
h=figure(h+1);
fignum=fignum+1;
plot(w/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2
numpoints],1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r--',...
  w(1:numpoints-2)/(2*pi),log10(abs(CORRHAD(ndx3d([aset_size aset_size (numpoints-
2)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b-.')
ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (Log10 of)'])
xlabel('Omega  (Hz)')
if titles
  title('Experimental FRF vs DK/DM/DC Corrected FRF')
end
grid on
v=axis;
hh=slegend('EXPERIMENTAL FRF','CORRECTED ANALYTIC FRF');
axes(hh);
if pswitch=='y'
  prtfig(fignum)
  delete(h)
end
clear CORRHAD

save plotnum fignum
clear
```

```
%%%%SSTCONF FILE FOR A SPATIALLY COMPLETE BEAM
beammdl        ;%beammdl is a cantilevered 20 element beam model
%simmdl        ;%simmdl is a masses and springs model
pswitch='n'    ;%%do we print?
titles=1       ;%%display titles
meters=1       ;%use progress meters
whitebg('black') ;%switch to black figure background
close
lowmode=1;
highmode=10;
static=1;%%%%%%%%%%%%%reduction method 0=Guyan,1=IRS,2=Extraction
%%define A set & O set%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aset=1:numdof;

%oset=2:2:numdof; %%A SPATIALLY INCOMPLETE BEAM%%%%%%%%%%

%oset=sort(oset); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

oset=[];  %%%%A SPATIALLY COMPLETE BEAM%%%%%%%%%%%%%%
save sstconf
sst
plotsst
```

CHAP4.M

```
%%%%SSTCONF FILE FOR A SPATIALLY INCOMPLETE BEAM
beammdl       ;%beammdl is a cantilevered 20 element beam model
%simmdl       ;%simmdl is a masses and springs model
pswitch='n'   ;%%do we print?
titles=1      ;%%display titles
meters=1      ;%use progress meters
whitebg('black') ;%switch to black figure background
close
lowmode=1;
highmode=10;

static=1;%%%%%%%%%%%%%reduction method 0=Guyan,1=IRS,2=Extraction
%%define A set & O set%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aset=1:numdof;
oset=2:2:numdof; %%A SPATIALLY INCOMPLETE BEAM%%%%%%%%%
oset=sort(oset); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

save sstconf
sst
plotsst

beammdl       ;%beammdl is a cantilevered 20 element beam model
%simmdl       ;%simmdl is a masses and springs model
pswitch='n'   ;%%do we print?
titles=1      ;%%display titles
meters=1      ;%use progress meters
whitebg('black') ;%switch to black figure background
close
lowmode=1;
highmode=10;

%%define A set & O set%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aset=1:numdof;
oset=2:2:numdof; %%A SPATIALLY INCOMPLETE BEAM%%%%%%%%%
oset=sort(oset); %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

static=2;
save sstconf
dofig4_3
```

CHAP5.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%%% COMPARE SINGLE MODE MATRIX AND INTEGRAL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% SOLUTIONS UNDER VARIOUS CONDITIONS     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
clear
closeall
load INT
whitebg('white')
clear L Z_ANAL c_anal c_exp conn h_exp h_anal_red k_anal k_exp
clear kexta kexto m_anal m_exp mexta mext0 temp_l_diags
clear true_damping true_mass true_stiffness z_anal z_anal_red z_exp
clear L Z_ANAL c_anal c_exp conn h_exp h_anal_red k_anal
clear kexta kexto m_anal mexta mext0 temp_l_diags

load H_EXP
load exp
cset=[5 7 9 11 13 15]
cset_rel=[3 4 5 6 7 8]
cset_size=length(cset)
mid_index=round(length(cset)/2);
fineness=200
ODZ1=[];
temp1=[];
odfreq=omegax(1);
center_freq=odfreq/(2*pi)
w1=odfreq;
odlength=2*pi;
odivisions=2;
lowerfreq=odfreq-15*odlength
upperfreq=odfreq+25*odlength
ow1=odfreq-.5*odlength:odlength/odivisions:odfreq+.5*odlength;
partition=ow1/(2*pi)
df=(odlength/odivisions)/(2*pi)
for count=1:length(ow1)
 z_anal_red=kstat-ow1(count)^2*mstat;
 h_anal_red=inv(z_anal_red);
 z_exp=k_exp+j*ow1(count)*c_exp-ow1(count)^2*m_exp;
 h_exp=inv(z_exp);
 h_exp=h_exp(aset,aset);
 hacc=h_anal_red(cset_rel,cset_rel);
 hxcc=h_exp(cset_rel,cset_rel);
 dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
 ODZ1=[ODZ1; dz];
 temp1=[temp1; eye(cset_size) -ow1(count)^2*eye(cset_size) +j*ow1(count)*eye(cset_size)];

end
%theleftdz=ODZ1
%thebigone=temp1
DKDMDC1=temp1\ODZ1;
ODK1=DKDMDC1(1:cset_size,:)
ODM1=DKDMDC1(cset_size+1:2*cset_size,:)
ODC1=DKDMDC1(2*cset_size+1:3*cset_size,:)
save ODM1 ODM1 DKDMDC1 ODZ1  ow1 w1
save ODK1 ODK1
save ODC1 ODC1

%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
load stat;
```

```
odcorrha=[];
w=lowerfreq:(upperfreq-lowerfreq)/fineness:upperfreq;
numpoints=length(w);
kcorrected=kstat;
mcorrected=mstat;
ccorrected=cstat;
kcorrected(cset_rel,cset_rel)=kcorrected(cset_rel,cset_rel)+ODK1;
mcorrected(cset_rel,cset_rel)=mcorrected(cset_rel,cset_rel)+ODM1;
ccorrected(cset_rel,cset_rel)=ccorrected(cset_rel,cset_rel)+ODC1;
for i=1:numpoints
 tempza=kcorrected+j*w(i)*ccorrected-w(i)^2*mcorrected;
 tempha=inv(tempza);
 odcorrha=[odcorrha tempha(:)];
end

uncorrha=[];
for i=1:numpoints
 tempza=kstat+j*w(i)*cstat-w(i)^2*mstat;
 tempha=inv(tempza);
 uncorrha=[uncorrha tempha(:)];
end

if meters
 waitbar_handle=waitbar(0,'Computing Experimental FRF');
else
 disp('Getting experimental FRF')
end
H_EXP=[];
for count=1:length(w)
 if meters
  waitbar(count/length(w));
 end
 z_exp=k_exp+j*w(count)*c_exp-w(count)^2*m_exp;
 h_exp=inv(z_exp);
 h_exp=h_exp(aset,aset);
 skyindex=1;
 for index1=1:aset_size
  for index2=index1:aset_size
   red_holder(skyindex)=h_exp(index1,index2);
   skyindex=skyindex+1;
  end
 end
 H_EXP=[H_EXP red_holder'];
end
if meters
 close(waitbar_handle)
end
clear tempha tempza kcorrected mcorrected ccorrected

figure(1);
plot(w/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 numpoints],...

   1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
   w/(2*pi),log10(abs(odcorrha(ndx3d([aset_size aset_size numpoints],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--',...

   w/(2*pi),log10(abs(uncorrha(ndx3d([aset_size aset_size numpoints],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:')

ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (log10 of)'])

xlabel('Omega (Hz)')
```

```
%title('Single Mode Corrected FRF (Mode 1) ')
grid
hh=slegend('Experimental FRF', 'Mode 1 Corrected FRF', 'Uncorrected Analytic FRF');

axes(hh)
print -dmfile fig5_1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%DECOMPOSE DZ into DM, DK, & DC   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% USING MATRIX TECHNIQUE          %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ODZ1=[];%%%%%%%%%
temp1=[];
odlength=2*pi;
ow1=omegax(1)-5*odlength:(omegax(2)-omegax(1)+10*odlength)/14:omegax(2)+5*odlength;

for count=1:length(ow1)
 z_anal_red=kstat-ow1(count)^2*mstat;
 h_anal_red=inv(z_anal_red);
 z_exp=k_exp+j*ow1(count)*c_exp-ow1(count)^2*m_exp;
 h_exp=inv(z_exp);
 h_exp=h_exp(aset,aset);
 hacc=h_anal_red(cset_rel,cset_rel);
 hxcc=h_exp(cset_rel,cset_rel);
 dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
 ODZ1=[ODZ1; dz];
 temp1=[temp1; eye(cset_size)  -ow1(count)^2*eye(cset_size) +j*ow1(count)*eye(cset_size)];

end

DKDMDC1=temp1\ODZ1;
ODK1=DKDMDC1(1:cset_size,:) ;
ODM1=DKDMDC1(cset_size+1:2*cset_size,:) ;
ODC1=DKDMDC1(2*cset_size+1:3*cset_size,:);
save ODM1 ODM1 DKDMDC1 ODZ1  ow1 w1
save ODK1 ODK1
save ODC1 ODC1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load stat;%%%%%%%%%
odcorrha=[];
w=omegax(1)-15*pi:(omegax(2)-omegax(1)+35*pi)/fineness:omegax(2)+20*pi;
numpoints=length(w);
kcorrected=kstat;
mcorrected=mstat;
ccorrected=cstat;
kcorrected(cset_rel,cset_rel)=kcorrected(cset_rel,cset_rel)+ODK1;
mcorrected(cset_rel,cset_rel)=mcorrected(cset_rel,cset_rel)+ODM1;
ccorrected(cset_rel,cset_rel)=ccorrected(cset_rel,cset_rel)+ODC1;
for i=1:numpoints
 tempza=kcorrected+j*w(i)*ccorrected-w(i)^2*mcorrected;
 tempha=inv(tempza);
 odcorrha=[odcorrha tempha(:)];
end

uncorrha=[];
for i=1:numpoints
 tempza=kstat+j*w(i)*cstat-w(i)^2*mstat;
 tempha=inv(tempza);
 uncorrha=[uncorrha tempha(:)];
```

```
    end

    if meters
      waitbar_handle=waitbar(0,'Computing Experimental FRF');
    else
      disp('Getting experimental FRF')
    end
    H_EXP=[];
    for count=1:length(w)
      if meters
        waitbar(count/length(w));
      end
      z_exp=k_exp+j*w(count)*c_exp-w(count)^2*m_exp;
      h_exp=inv(z_exp);
      h_exp=h_exp(aset,aset);
      skyindex=1;
      for index1=1:aset_size
        for index2=index1:aset_size
          red_holder(skyindex)=h_exp(index1,index2);
          skyindex=skyindex+1;
        end
      end
      H_EXP=[H_EXP red_holder'];
    end
    if meters
      close(waitbar_handle)
    end
    clear tempha tempza kcorrected mcorrected ccorrected

    figure(2);
    plot(w/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 numpoints],...

        1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
        w/(2*pi),log10(abs(odcorrha(ndx3d([aset_size aset_size numpoints],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--',...

        w/(2*pi),log10(abs(uncorrha(ndx3d([aset_size aset_size numpoints],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:')

    ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') in/lbf (log10 of)'])

    xlabel('Omega (Hz)')
    %title('Single Mode Corrected FRF (Mode 1) ')
    grid
    hh=slegend('Experimental FRF', 'Single Mode Corrected FRF', 'Uncorrected Analytic FRF');

    axes(hh)
    print -dmfile fig5_2

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%determine ODZ into ODM, ODK, & ODC %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    plotw=lowerfreq:(upperfreq-lowerfreq)/fineness:upperfreq;
    lenctrl=[25 10 1];
    odivisions=3;
    for ind=1:length(lenctrl)
      odlength=lenctrl(ind)*2*pi;
      ODZ=[];
      temp=[];
      ow=odfreq-.5*odlength:odlength/odivisions:odfreq+.5*odlength;
      for count=1:length(ow)
        z_anal_red=kstat-ow(count)^2*mstat;
```

```
    h_anal_red=inv(z_anal_red);
    z_exp=k_exp+j*ow(count)*c_exp-ow(count)^2*m_exp;
    h_exp=inv(z_exp);
    h_exp=h_exp(aset,aset);
    hacc=h_anal_red(cset_rel,cset_rel);
    hxcc=h_exp(cset_rel,cset_rel);
    dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
    ODZ=[ODZ; dz];
    temp=[temp; eye(cset_size)  -ow(count)^2*eye(cset_size) +j*ow(count)*eye(cset_size)];

  end

  DKDMDC=temp\ODZ;
  ODK=DKDMDC(1:cset_size,:) ;
  ODM=DKDMDC(cset_size+1:2*cset_size,:) ;
  ODC=DKDMDC(2*cset_size+1:3*cset_size,:);
  clear hacc hxcc h_exp z_exp z_anal_red h_anal_red dz
  save ODM ODM DKDMDC ODZ  ow
  save ODK ODK
  save ODC ODC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%if meters%%%%
   waitbar_handle=waitbar(0,'Computing Experimental FRF');
  else
   disp('Getting experimental FRF')
  end
  H_EXP=[];
  for count=1:length(plotw)
   if meters
    waitbar(count/length(plotw));
   end
   z_exp=k_exp+j*plotw(count)*c_exp-plotw(count)^2*m_exp;
   h_exp=inv(z_exp);
   h_exp=h_exp(aset,aset);
   skyindex=1;
   for index1=1:aset_size
    for index2=index1:aset_size
     red_holder(skyindex)=h_exp(index1,index2);
     skyindex=skyindex+1;
    end
   end
  H_EXP=[H_EXP red_holder'];
 end
 if meters
  close(waitbar_handle)
 end

 odcorrha=[];
 kcorrected=kstat;
 mcorrected=mstat;
 ccorrected=cstat;
 kcorrected(cset_rel,cset_rel)=kcorrected(cset_rel,cset_rel)+ODK;
 mcorrected(cset_rel,cset_rel)=mcorrected(cset_rel,cset_rel)+ODM;
 ccorrected(cset_rel,cset_rel)=ccorrected(cset_rel,cset_rel)+ODC;
 for i=1:length(plotw)
  tempza=kcorrected+j*plotw(i)*ccorrected-plotw(i)^2*mcorrected;
  tempha=inv(tempza);
  odcorrha=[odcorrha tempha(:)];
 end
 if ind==1
```

```
    plot1=odcorrha;
   elseif ind==2
    plot2=odcorrha;
   elseif ind==3
    plot3=odcorrha;
   else
    plot4=odcorrha;
  end
 end

figure(3);
plot(plotw/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(plotw)],...

    1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
    plotw/(2*pi), log10(abs(plot1(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--
',...
    plotw/(2*pi), log10(abs(plot2(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:',...
    plotw/(2*pi), log10(abs(plot3(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'k-.')

ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (log10 of)'])

xlabel('Omega (Hz)')
grid on
%title('EXPERMENTAL FRF AND CORRECTED FRFS USING MODE 5 SOLUTIONS')
hh=legend('Exp FRF', ...
     [num2str(lenctrl(1)),' Hz Bandwidth Corrected FRF'], ...
     [num2str(lenctrl(2)),' Hz Bandwidth Corrected FRF'], ...
     [num2str(lenctrl(3)),' Hz Bandwidth Corrected FRF']) ;
axes(hh)
print -dmfile fig5_3

odlength=1*2*pi;
countctrl=[3 10 50 200] ;
for ind=1:length(countctrl)
 odivisions=countctrl(ind)-1;
 ODZ=[];
 temp=[];
 ow=odfreq-.5*odlength:odlength/odivisions:odfreq+.5*odlength;
 for count=1:length(ow)
  z_anal_red=kstat-ow(count)^2*mstat;
  h_anal_red=inv(z_anal_red);
  z_exp=k_exp+j*ow(count)*c_exp-ow(count)^2*m_exp;
  h_exp=inv(z_exp);
  h_exp=h_exp(aset,aset);
  hacc=h_anal_red(cset_rel,cset_rel);
  hxcc=h_exp(cset_rel,cset_rel);
  dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
  ODZ=[ODZ; dz];
  temp=[temp; eye(cset_size)  -ow(count)^2*eye(cset_size) +j*ow(count)*eye(cset_size)];

 end

DKDMDC=temp\ODZ;
ODK=DKDMDC(1:cset_size,:) ;
ODM=DKDMDC(cset_size+1:2*cset_size,:) ;
ODC=DKDMDC(2*cset_size+1:3*cset_size,:);
clear hacc hxcc h_exp z_exp z_anal_red h_anal_red dz
save ODM ODM DKDMDC ODZ  ow
save ODK ODK
```

```
 save ODC ODC
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
 odcorrha=[];
 kcorrected=kstat;
 mcorrected=mstat;
 ccorrected=cstat;
 kcorrected(cset_rel,cset_rel)=kcorrected(cset_rel,cset_rel)+ODK;
 mcorrected(cset_rel,cset_rel)=mcorrected(cset_rel,cset_rel)+ODM;
 ccorrected(cset_rel,cset_rel)=ccorrected(cset_rel,cset_rel)+ODC;
 for i=1:length(plotw)
  tempza=kcorrected+j*plotw(i)*ccorrected-plotw(i)^2*mcorrected;
  tempha=inv(tempza);
  odcorrha=[odcorrha tempha(:)];
 end
 if ind==1
  plot1=odcorrha;
 elseif ind==2
  plot2=odcorrha;
 elseif ind==3
  plot3=odcorrha;
 else
  plot4=odcorrha;
 end
end

figure(4);
plot(plotw/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(plotw)],...

   1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
   plotw/(2*pi), log10(abs(plot1(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--
',...
   plotw/(2*pi), log10(abs(plot2(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:',...
   plotw/(2*pi), log10(abs(plot3(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'k-
.',...
   plotw/(2*pi), log10(abs(plot4(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'m:')

ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),') in/lbf (log10 of)'])

xlabel('Omega (Hz)')
grid on
%title('EXPERMENTAL FRF AND CORRECTED FRFS USING MODE 1 SOLUTIONS')
hh=legend('Exp FRF', ...
     [num2str(countctrl(1)),' Pts Corrected FRF'], ...
     [num2str(countctrl(2)),' Pts Corrected FRF'], ...
     [num2str(countctrl(3)),' Pts Corrected FRF'], ...
     [num2str(countctrl(4)),' Pts Corrected FRF']);
axes(hh)
print -dmfile fig5_4

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%DECOMPOSE DZ into DM, DK, & DC %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% USING INTEGRAL TECHNIQUE    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%dw=2*pi;%%%%%%
%plotw=lowerfreq:dw:upperfreq;
lenctrl=[25 10 1];
odivisions=3;
for ind=1:length(lenctrl)
```

```
odlength=lenctrl(ind)*2*pi;
owl=odfreq-.5*odlength:odlength/(odivisions):odfreq+.5*odlength;
W1=owl;
%%compute integrals  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%weight=ones(size(owl))*(omegax(1));
weight=owl;
DZ_R=[];
DZ_I=[];
for count=1:length(owl)
 z_anal_red=kstat/(j*owl(count))+j*owl(count)*mstat;
 h_anal_red=inv(z_anal_red);
 z_exp=k_exp/(j*owl(count))+c_exp/(j*owl(count))+j*owl(count)*m_exp;
 h_exp=inv(z_exp);
 h_exp=h_exp(aset,aset);
 hacc=h_anal_red(cset_rel,cset_rel);
 hxcc=h_exp(cset_rel,cset_rel);
 dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
 dz_r=real(dz);
 dz_i=imag(dz);
 DZ_R=[DZ_R dz_r(:)];
 DZ_I=[DZ_I dz_i(:)];
end

[INTK INTM INTC]=intsub(DZ_I,DZ_R,owl,W1,1./(j*owl),cset_size);
odcorrha=[];
load stat
kcorrected=kstat;
mcorrected=mstat;
ccorrected=cstat;
kcorrected(cset_rel,cset_rel)=kcorrected(cset_rel,cset_rel)+INTK;
mcorrected(cset_rel,cset_rel)=mcorrected(cset_rel,cset_rel)+INTM;
ccorrected(cset_rel,cset_rel)=ccorrected(cset_rel,cset_rel)+INTC;
for i=1:length(plotw)
 tempza=kcorrected+j*plotw(i)*ccorrected-plotw(i)^2*mcorrected;
 tempha=inv(tempza);
 odcorrha=[odcorrha tempha(:)];
end
if ind==1
 intplot1=odcorrha;
 save INTM1 INTM
 save INTK1 INTK
 save INTC1 INTC
elseif ind==2
 intplot2=odcorrha;
 save INTM2 INTM
 save INTK2 INTK
 save INTC2 INTC
elseif ind==3
 intplot3=odcorrha;
 save INTM3 INTM
 save INTK3 INTK
 save INTC3 INTC
else
 intplot4=odcorrha;
 save INTM4 INTM
 save INTK4 INTK
 save INTC4 INTC
end
end
```

```
if meters
  waitbar_handle=waitbar(0,'Computing Experimental FRF');
else
  disp('Getting experimental FRF')
end
H_EXP=[];
for count=1:length(plotw)
  if meters
    waitbar(count/length(plotw));
  end
  z_exp=k_exp+j*plotw(count)*c_exp-plotw(count)^2*m_exp;
  h_exp=inv(z_exp);
  h_exp=h_exp(aset,aset);
  skyindex=1;
  for index1=1:aset_size
    for index2=index1:aset_size
      red_holder(skyindex)=h_exp(index1,index2);
      skyindex=skyindex+1;
    end
  end
  H_EXP=[H_EXP red_holder'];
end
if meters
  close(waitbar_handle)
end

figure(5);
plot(plotw/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(plotw)],...

     1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
     plotw/(2*pi), log10(abs(intplot1(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--',...
     plotw/(2*pi), log10(abs(intplot2(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:',...
     plotw/(2*pi), log10(abs(intplot3(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'k-.')
ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (log10 of)'])

xlabel('Omega (Hz)')
grid on
%title('EXPERMENTAL FRF AND CORRECTED FRFS USING MODE 1 INT SOLUTIONS')
hh=legend('Exp FRF', ...
     [num2str(lenctrl(1)),' Hz Bandwidth Corrected FRF'],...
     [num2str(lenctrl(2)),' Hz Bandwidth Corrected FRF'],...
     [num2str(lenctrl(3)),' Hz Bandwidth Corrected FRF']);
axes(hh)
print -dmfile fig5_5

odlength=2*pi;
countctrl=[3 10 50 200];
for ind=1:length(countctrl)
  odivisions=countctrl(ind);
  ow1=odfreq-.5*odlength:odlength/(odivisions-1):odfreq+.5*odlength;
  W1=ow1;
  %%compute integrals   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %weight=ones(size(ow1))*(omegax(1));
  weight=ow1;
  if ind == 1
    ow1
    df=(odlength/(odivisions-1))/(2*pi)
```

```
end
%% compute integrals  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DZ_R=[];
DZ_I=[];
for count=1:length(ow1)
 z_anal_red=kstat/(j*ow1(count))+j*ow1(count)*mstat;
 h_anal_red=inv(z_anal_red);
 z_exp=k_exp/(j*ow1(count))+c_exp/(j*ow1(count))+j*ow1(count)*m_exp;
 h_exp=inv(z_exp);
 h_exp=h_exp(aset,aset);
 hacc=h_anal_red(cset_rel,cset_rel);
 hxcc=h_exp(cset_rel,cset_rel);
 dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
 dz_r=real(dz);
 dz_i=imag(dz);
 DZ_R=[DZ_R dz_r(:)];
 DZ_I=[DZ_I dz_i(:)];
end
if ind == 1
 DZ_R;
 DZ_I;
 for i=1:length(ow1)
  dztemp1=DZ_R(ndx3d([1 cset_size*(cset_size+1)/2 length(ow1)],1,1:cset_size*(cset_size+1)/2,i));

  dztemp2=DZ_I(ndx3d([1 cset_size*(cset_size+1)/2 length(ow1)],1,1:cset_size*(cset_size+1)/2,i));

  skyindex=1;
  for index1=1:cset_size
   for index2=index1:cset_size
    dz1(index1,index2)=dztemp1(skyindex);
    dz1(index2,index1)=dztemp1(skyindex);
    dz2(index1,index2)=dztemp2(skyindex);
    dz2(index2,index1)=dztemp2(skyindex);
    skyindex=skyindex+1;
   end
  end
  dz1
  dz2
 end

end
[INTK INTM INTC]=intsub(DZ_I,DZ_R,ow1,W1,1./(j*ow1),cset_size);
if ind == 1
 [INTK INTM INTC]=indsub(DZ_I,DZ_R,ow1,W1,1./(j*ow1),cset_size);
 INTM
 INTK
 INTC
end

%%corrected%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%stat
kcorrected=kstat;
mcorrected=mstat;
ccorrected=cstat;
kcorrected(cset_rel,cset_rel)=kcorrected(cset_rel,cset_rel)+INTK;
mcorrected(cset_rel,cset_rel)=mcorrected(cset_rel,cset_rel)+INTM;
ccorrected(cset_rel,cset_rel)=ccorrected(cset_rel,cset_rel)+INTC;
for i=1:length(plotw)
 tempza=kcorrected+j*plotw(i)*ccorrected-plotw(i)^2*mcorrected;
 tempha=inv(tempza);
```

```
   odcorrha=[odcorrha tempha(:)];
  end
  if ind==1
   intplot1=odcorrha;
   save INTM1 INTM
   save INTK1 INTK
   save INTC1 INTC
  elseif ind==2
   intplot2=odcorrha;
   save INTM2 INTM
   save INTK2 INTK
   save INTC2 INTC
  elseif ind==3
   intplot3=odcorrha;
   save INTM3 INTM
   save INTK3 INTK
   save INTC3 INTC
  else
   intplot4=odcorrha;
   save INTM4 INTM
   save INTK4 INTK
   save INTC4 INTC
  end
 end

 if meters
  waitbar_handle=waitbar(0,'Computing Experimental FRF');
 else
  disp('Getting experimental FRF')
 end
 H_EXP=[];
 for count=1:length(plotw)
  if meters
   waitbar(count/length(plotw));
  end
  z_exp=k_exp+j*plotw(count)*c_exp-plotw(count)^2*m_exp;
  h_exp=inv(z_exp);
  h_exp=h_exp(aset,aset);
  skyindex=1;
  for index1=1:aset_size
   for index2=index1:aset_size
    red_holder(skyindex)=h_exp(index1,index2);
    skyindex=skyindex+1;
   end
  end
  H_EXP=[H_EXP red_holder'];
 end
 if meters
  close(waitbar_handle)
 end

 figure(6);
 plot(plotw/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(plotw)],...

    1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
    plotw/(2*pi), log10(abs(intplot1(ndx3d([aset_size aset_size
 length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--',...
    plotw/(2*pi), log10(abs(intplot2(ndx3d([aset_size aset_size
 length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:',...
```

```
    plotw/(2*pi), log10(abs(intplot3(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'k-.',...
    plotw/(2*pi), log10(abs(intplot4(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'m:')
ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (log10 of)'])

xlabel('Omega (Hz)')
grid on
%title('EXPERMENTAL FRF AND CORRECTED FRFS USING MODE 1 INT SOLUTIONS')
hh=slegend('Exp FRF', ...
     [num2str(countctrl(1)),' Pts Corrected FRF'], ...
     [num2str(countctrl(2)),' Pts Corrected FRF'], ...
     [num2str(countctrl(3)),' Pts Corrected FRF'], ...
     [num2str(countctrl(4)),' Pts Corrected FRF']);
axes(hh);
print -dmfile fig5_6
figure(7);
plot(plotw/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(plotw)],...

    1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
    plotw/(2*pi), log10(abs(plot1(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--
',...
    plotw/(2*pi), log10(abs(intplot1(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:')
ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (log10 of)'])

xlabel('Omega (Hz)')
if titles
 % title('EXPERMENTAL FRF AND CORRECTED FRFS USING MODE 1 OD&INTGRL SOLTNS')
end
grid on
hh=legend('Exp FRF', ...
     [num2str(lenctrl(1)),' Hz Bandwidth MAT Corrected FRF'], ...
     [num2str(lenctrl(1)),' Hz Bandwidth INT Corrected FRF']);
axes(hh)
if pswitch=='y'
  print -dcdjcolor
end
print -dmfile fig5_7

figure(8);
plot(plotw/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(plotw)],...

    1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
    plotw/(2*pi), log10(abs(plot2(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--
',...
    plotw/(2*pi), log10(abs(intplot2(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:')
ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (log10 of)'])

xlabel('Omega (Hz)')
if titles
  %title('EXPERMENTAL FRF AND CORRECTED FRFS USING MODE 1 OD&INTGRL SOLTNS')
end
grid on
hh=legend('Exp FRF', ...
     [num2str(lenctrl(2)),' Hz Bandwidth MAT Corrected FRF'], ...
     [num2str(lenctrl(2)),' Hz Bandwidth INT Corrected FRF']);
axes(hh)
if pswitch=='y'
```

```
  print -dcdjcolor
 end
 print -dmfile fig5_8

 figure(9);
 plot(plotw/(2*pi),log10(abs(H_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(plotw)],...

     1,skyred(cset_rel(mid_index),cset_rel(mid_index)),':')))),'r-',...
     plotw/(2*pi), log10(abs(plot3(ndx3d([aset_size aset_size length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'g--
',...
     plotw/(2*pi), log10(abs(intplot3(ndx3d([aset_size aset_size
length(plotw)],cset_rel(mid_index),cset_rel(mid_index),':')))),'b:')
ylabel(['H(',int2str(cset(mid_index)),',',int2str(cset(mid_index)),')  in/lbf (log10 of)'])

xlabel('Omega (Hz)')
if titles
  %title('EXPERMENTAL FRF AND CORRECTED FRFS USING MODE 1 OD&INTGRL SOLTNS')
end
grid on
hh=legend('Exp FRF', ...
       [num2str(lenctrl(3)),' Hz Bandwidth MAT Corrected FRF'], ...
       [num2str(lenctrl(3)),' Hz Bandwidth INT Corrected FRF']);
axes(hh)
if pswitch=='y'
  print -dcdjcolor
end
print -dmfile fig5_9
```

```
clear
odivisions=3
startloop=1
skiploop=1
endloop=4
cd incomp1
save odconf odivisions startloop skiploop endloop
chap6_1
chap6_2
cd ..
clear
odivisions=1
startloop=3
skiploop=1
endloop=4
cd comp1
save odconf odivisions startloop skiploop endloop
chap6_1
chap6_3
```

CHAP6.M

```
%%%CHAP6_1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%%%Decompose DZ into DM, DK, & DC%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%using matrix formulation%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
clc
clear
closeall
load INT
%if pswitch=='y'
whitebg('white')
close
%end
h=0;
fignum=1;
use_antires=1;
titles=0;
mid_index=round(length(cset)/2);

clear L Z_ANAL c_exp conn h_anal_red
clear kexto m_exp mexto temp_1_diags
clear true_damping true_mass true_stiffness z_anal z_anal_red z_exp
clear L Z_ANAL c_exp conn h_anal_red
clear kexta kexto mext0 temp_1_diags

%%%%%%%%%%%%%%%%%%%%FORCE THE CSET%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%SPATIALLY INCOMPLETE%%%%%%%%%%%%%%%%%%%
if complete
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%SPATIALLY COMPLETE%%%%%%%%%%%%%%%%%%%
  cset=[4 5 6 7 8 9 10 11]%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_rel=cset%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_size=length(cset)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
%%%%%%%%%%%%SPATIALLY INCOMPLETE%%%%%%%%%%%%%%%%%%%%
  cset=[1 3 5 7 9 11 13 15]%%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_rel=[1 2 3 4 5 6 7 8]%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_size=length(cset)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


load exp
odivisions=3 ;
load odconf
nummodes=8   ;
odlength=2.001*pi ;
firsttime=1;
dw=odlength/(odivisions+1);
dow=(freqtop-freqbottom)/(fineness-1);
owref=freqbottom:dow:freqtop;

if firsttime
  if meters
    waitbar_handle=waitbar(0,'Computing Experimental FRF');
  else
    disp('Getting experimental FRF')
  end
```

```
    H_T_EXP=[];
    for count=1:length(owref)
      if meters
        waitbar(count/length(owref));
      end
      z_exp=k_exp+j*owref(count)*c_exp-owref(count)^2*m_exp;
      h_exp=inv(z_exp);
      h_exp=h_exp(aset,aset);
   %  skyindex=1;
   %  for index1=1:aset_size
   %    for index2=index1:aset_size
   %      red_holder(skyindex)=h_exp(index1,index2);
   %      skyindex=skyindex+1;
   %    end
   %  end
   %  H_T_EXP=[H_T_EXP red_holder'];
      H_T_EXP=[H_T_EXP h_exp(:)];
    end
    if meters
      close(waitbar_handle)
    end
    save H_T_EXP H_T_EXP
    clear H_T_EXP
    %load H_ANAL_RED;
    %uncorrha=H_ANAL_RED;
    %clear H_ANAL_RED
    uncorrha=[];
    if meters
      waitbar_handle=waitbar(0,'Computing Uncorrected FRF');
    else
      disp('Getting Uncorrected FRF')
    end
    load stat
    for i=1:length(owref)
      if meters
        waitbar(i/length(owref));
      end
      tempza=kstat+j*owref(i)*cstat-owref(i)^2*mstat;
      tempha=inv(tempza);
      uncorrha=[uncorrha tempha(:)];
    end
    if meters
      close(waitbar_handle)
    end
    save uncorrha uncorrha
    clear uncorrha
  end
  load H_T_EXP
  temp=[];
  for index=1:nummodes
    for coord=1:aset_size
      if index == 1
        antiresfreqs=find(owref>0 & owref<omegax(index));
      else
        antiresfreqs=find(owref>omegax(index-1) & owref<omegax(index));
      end
      antires=min(log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
        coord,coord,antiresfreqs)))));
      whre=find(log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
        coord,coord,antiresfreqs))))==antires);
```

```
        temp=owref(antiresfreqs) ;
        antiresfreq(coord,index)=temp(whre(1));
      end
    end
    temp=[];
    for index=1:nummodes
      resfreqs=find(owref>omegax(index)-dow & owref<omegax(index)+dow);
      res=max(log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
        cset_rel(1),cset_rel(1),resfreqs)))));
      whre=find(log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
        cset_rel(1),cset_rel(1),resfreqs))))==res);
      temp=owref(resfreqs) ;
      resfreq(index)=temp(whre(1));
    end
    clear H_T_EXP

    %clear omegax
    %omegax=resfreq;

    for loopindex=startloop:skiploop:endloop
      ODZ=[];
      temp=[];
      rdiag=[];
      ow=[];
      lowwer=owref(1)-.5*odlength;
      uppper=owref(1)+.5*odlength ;
    % if odivisions == 1
    %    ow=[ow owref(1)]
    % else
    %    ow=[ow lowwer+dw:dw:uppper-dw];
    % end
    % rdiag=[rdiag ones(1,odivisions*cset_size).*owref(1)];


      for index=1:loopindex
    %      lowwer=omegax(index)-.5*odlength ;
    %      uppper=omegax(index)+.5*odlength ;

        lowwer=resfreq(index)-.5*odlength ;
        uppper=resfreq(index)+.5*odlength ;
        ow=[ow lowwer+dw:dw:uppper-dw];
        rdiag=[rdiag ones(1,odivisions*cset_size).*resfreq(index)];
        if use_antires

          for coord=1:aset_size
            lowwer=antiresfreq(coord,index)-.5*odlength;
            uppper=antiresfreq(coord,index)+.5*odlength;
            size(lowwer+dw:dw:uppper-dw)
            ow=[ow lowwer+dw:dw:uppper-dw]
            rdiag=[rdiag ones(1,odivisions*cset_size).*resfreq(index)];
          end
        end
      end
      R=diag(rdiag);
      load stat
      if meters
        waitbar_handle=waitbar(0,'Computing DK/DM/DC');
      else
        disp('Getting DK/DM/DC')
      end
```

```
 for count=1:length(ow)
   if meters
     waitbar(count/length(ow));
   end
   z_anal_red=kstat-ow(count)^2*mstat;
   h_anal_red=inv(z_anal_red);
   z_exp=k_exp+j*ow(count)*c_exp-ow(count)^2*m_exp;
   h_exp=inv(z_exp);
   h_exp=h_exp(aset,aset);
   hacc=h_anal_red(cset_rel,cset_rel);
   hxcc=h_exp(cset_rel,cset_rel);
   dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
   ODZ=[ODZ; dz];
   temp=[temp; eye(cset_size) -ow(count)^2*eye(cset_size) +j*ow(count)*eye(cset_size)];
 end
 if meters
   close(waitbar_handle)
 end
 DKDMDC1=inv(temp'*inv(R)*temp)*temp'*inv(R)*ODZ;
%DKDMDC1=temp\ODZ ;
 ODK1=DKDMDC1(1:cset_size,:) ;
 ODM1=DKDMDC1(cset_size+1:2*cset_size,:) ;
 ODC1=DKDMDC1(2*cset_size+1:3*cset_size,:) ;

 clear hacc hxcc h_exp z_exp z_anal_red h_anal_red dz
 clear temp ODZ


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
 load stat;
 odcorrha=[];
 kstat(cset_rel,cset_rel)=kstat(cset_rel,cset_rel)+ODK1;
 mstat(cset_rel,cset_rel)=mstat(cset_rel,cset_rel)+ODM1;;
 cstat(cset_rel,cset_rel)=cstat(cset_rel,cset_rel)+ODC1;

 if meters
   waitbar_handle=waitbar(0,'Computing Corrected FRF');
 else
   disp('Getting DK/DM/DC')
 end
 for i=1:length(owref)
   if meters
     waitbar(i/length(owref));
   end
   tempza=kstat+j*owref(i)*cstat-owref(i)^2*mstat;
   tempha=inv(tempza);
   odcorrha=[odcorrha tempha(:)];
 end
 if meters
   close(waitbar_handle)
 end
 if loopindex == 1
   odcorrha1=odcorrha;
   save od1 odcorrha1
   clear odcorrha1
 elseif loopindex == 2
   odcorrha2=odcorrha;
   save od2 odcorrha2
```

```
        clear odcorrha2
      elseif loopindex == 3
        odcorrha3=odcorrha;
        save od3 odcorrha3
        clear odcorrha3
      elseif loopindex == 4
        odcorrha4=odcorrha;
        save od4 odcorrha4
        clear odcorrha4
      elseif loopindex == 5
        odcorrha5=odcorrha;
        save od5 odcorrha5
        clear odcorrha5
      end

      clear tempha tempza kcorrected mcorrected ccorrected
      clear ODM1 ODC1 DKDMDC1 ODZ temp kstat mstat

      load uncorrha
      load H_T_EXP
      if complete
        plotwhichcoord=[cset(mid_index)];
      else
        plotwhichcoord=[cset(mid_index)];
      end
      for plotindex=1:length(plotwhichcoord)
        which_coord=plotwhichcoord(plotindex);
        h=figure(h+1);
        fignum=fignum+1  ;
        subplot(2,1,1);
        plot(owref/(2*pi),...
          log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
          which_coord,which_coord,':')))),'r-',...
        owref/(2*pi),...
          log10(abs(odcorrha(ndx3d([aset_size aset_size length(owref)],...
          which_coord,which_coord,':')))),'g--',...
        owref/(2*pi),...
          log10(abs(uncorrha(ndx3d([aset_size aset_size length(owref)],...
          which_coord,which_coord,':')))),'b-.')
        ylabel(['H(',int2str(aset(which_coord)),...
          ',',int2str(aset(which_coord)),')  in/lbf (log10 of)'])
        hold on
        v=axis;
        %xlabel('Omega (Hz)')
        if titles
          if use_antires
            title([int2str(odivisions),' PT MATRIX SOLTNS WITH ANTIRESONANCES WEIGHTED LEFT TO RIGHT'])
          else
            title([int2str(odivisions),' PT MATRIX SOLTNS WEIGHTED LEFT TO RIGHT'])
          end
        end
        plot(resfreq(1:loopindex)/(2*pi),...
          ones(1,loopindex)*(v(3)+abs(v(4)-v(3))*.98),'k+')
    %     if use_antires
    %     %plot(diag(antiresfreq(1:loopindex,1:loopindex))/(2*pi),ones(1,loopinde%x)*(v(3)+abs(v(4)-v(3))*.98),'k*')
    %     end
        grid on
        hh=legend('Experimental FRF','Corrected MAT Anal FRF','Uncorrected Anal FRF','Included Modes');
        axes(hh)
        hold off
```

```
      %if pswitch=='y'
      %  prtfig1(fignum)
      %  delete(h)
%     % else
%       delete(h)
    % end
    subplot(2,1,2)
    thetop=find(abs(owref-ow(length(ow))*1.5*ones(1,length(owref)))==min(abs(owref-
ow(length(ow))*1.5*ones(1,length(owref)))));

    plot(owref(1:thetop)/(2*pi),...
       log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
       which_coord,which_coord,1:thetop)))),'r-',...
    owref(1:thetop)/(2*pi),...
       log10(abs(odcorrha(ndx3d([aset_size aset_size length(owref)],...
       which_coord,which_coord,1:thetop)))),'g--',...
    owref(1:thetop)/(2*pi),...
       log10(abs(uncorrha(ndx3d([aset_size aset_size length(owref)],...
       which_coord,which_coord,1:thetop)))),'b-.')
    ylabel(['H(',int2str(aset(which_coord)),',',int2str(aset(which_coord)),')  in/lbf (log10 of)'])
    hold on
    v=axis;
    xlabel('Omega (Hz)')
    if titles
      if use_antires
        title([int2str(odivisions),' PT MATRIX SOLTNS WITH ANTIRESONANCES WEIGHTED LEFT TO RIGHT'])
      else
        title([int2str(odivisions),' PT MATRIX SOLTNS WEIGHTED LEFT TO RIGHT'])
      end
    end
    plot(resfreq(1:loopindex)/(2*pi),ones(1,loopindex)*(v(3)+abs(v(4)-v(3))*.98),'k+')
%      if use_antires
%    %plot(antiresfreq(which_coord,1:loopindex)/(2*pi),ones(1,loopindex)*(v(%3)+abs(v(4)-v(3))*.98),'k*')
%      end
    grid on
    %hh=legend('Experimental FRF','Corrected MAT Anal FRF','Uncorrected Anal FRF','Included Modes');
    %axes(hh)
    hold off
   if loopindex == 2
     print -dcdjcolor
     print -dmfile fig6_1
   elseif loopindex == 3
     if odivisions == 1
       print -dcdjcolor
       if complete
         print -dmfile fig6_9
       else
         print -dmfile fig6_6
       end
     end
   elseif loopindex == 4
     print -dcdjcolor
     print -dmfile fig6_2
   end
 %   if pswitch=='y'
 %     prtfig1(figmm)
 %     delete(h)
 %   else
 %     delete(h)
 %   end
```

```
    end
    delete(h)
    clear H_T_EXP
    clear odcorrha
    clear uncorrha
end
```

```
%%%%CHAP6_2.M%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%%% Decompose DZ into DM, DK, & DC %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% using matrix formulation     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
clc
clear
closeall
load INT
whitebg('white')
close
h=0;
fignum=1;
use_antires=1
titles=0
mid_index=round(length(cset)/2);

clear L Z_ANAL c_exp conn h_anal_red
clear kexto m_exp mexto temp_1_diags
clear true_damping true_mass true_stiffness z_anal z_anal_red z_exp
clear L Z_ANAL c_exp conn h_anal_red
clear kexta kexto mext0 temp_I_diags

%%%%%%%%%%%%%%%%%%%%FORCE THE CSET%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%SPATIALLY INCOMPLETE%%%%%%%%%%%%%%%%%%%%
if complete
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%SPATIALLY COMPLETE%%%%%%%%%%%%%%%%%%
  cset=[4 5 6 7 8 9 10 11]%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_rel=cset%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_size=length(cset)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
else
%%%%%%%%%%%%%SPATIALLY INCOMPLETE%%%%%%%%%%%%%%%%%%%%
  cset=[1 3 5 7 9 11 13 15]%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_rel=[1 2 3 4 5 6 7 8]%%%%%%%%%%%%%%%%%%%%%%%%%
  cset_size=length(cset)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%SPATIALLY COMPLETE%%%%%%%%%%%%%%%%%%
%cset=[7 8 9 10 11 12 13 14]%%%%%%%%%%%%%%%%%%%%%%%%
%cset_rel=cset%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cset_size=length(cset)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load exp
odivisions=3  ;
load odconf
nummodes=9   ;
odlength=2*pi ;
firsttime=1;
dw=odlength/(odivisions+1);
dow=(freqtop-freqbottom)/(fineness-1);
owref=freqbottom:dow:freqtop;

if firsttime
  if meters
```

```
    waitbar_handle=waitbar(0,'Computing Experimental FRF');
  else
    disp('Getting experimental FRF')
  end

  H_T_EXP=[];
  for count=1:length(owref)
    if meters
      waitbar(count/length(owref));
    end
    z_exp=k_exp+j*owref(count)*c_exp-owref(count)^2*m_exp;
    h_exp=inv(z_exp);
    h_exp=h_exp(aset,aset);
%   skyindex=1;
%   for index1=1:aset_size
%     for index2=index1:aset_size
%       red_holder(skyindex)=h_exp(index1,index2);
%       skyindex=skyindex+1;
%     end
%   end
%   H_T_EXP=[H_T_EXP red_holder'];
    H_T_EXP=[H_T_EXP h_exp(:)];
  end


  if meters
    close(waitbar_handle)
  end


  save H_T_EXP H_T_EXP
  clear H_T_EXP
end
%load H_ANAL_RED;
%uncorrha=H_ANAL_RED;
%clear H_ANAL_RED
if firsttime
  uncorrha=[];
  if meters
    waitbar_handle=waitbar(0,'Computing Uncorrected FRF');
  else
    disp('Getting Uncorrected FRF')
  end

  load stat
  for i=1:length(owref)
    if meters
      waitbar(i/length(owref));
    end
    tempza=kstat+j*owref(i)*cstat-owref(i)^2*mstat;
    tempha=inv(tempza);
    uncorrha=[uncorrha tempha(:)];
  end

  if meters
    close(waitbar_handle)
  end

  save uncorrha uncorrha
  clear uncorrha
```

```
end

load H_T_EXP
for index=1:nummodes
  for coord=1:aset_size
    if index == 1
      antiresfreqs=find(owref>0 & owref<omegax(index));
    else
      antiresfreqs=find(owref>omegax(index-1) & owref<omegax(index));
    end
    antires=min(log10(abs(H_T_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(owref)],...
      1,skyred(coord,coord),antiresfreqs)))));
    whre=find(log10(abs(H_T_EXP(ndx3d([1 aset_size*(aset_size+1)/2 length(owref)],...
      1,skyred(coord,coord),antiresfreqs))))==antires);
    temp=owref(antiresfreqs) ;
    antiresfreq(coord,index)=temp(whre);
  end
end

for index=1:nummodes
  resfreqs=find(owref>omegax(index)-dow & owref<omegax(index)+dow);
  res=max(log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
    cset_rel(1),cset_rel(1),resfreqs)))));
  whre=find(log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
    cset_rel(1),cset_rel(1),resfreqs))))==res);
  temp=owref(resfreqs) ;
  resfreq(index)=temp(whre);
end

clear H_T_EXP
%clear omegax
%omegax=resfreq;

for loopindex=startloop:skiploop:endloop
  subdivisions=9;          % should be odd for simpson's rule
  intlength=2*pi;          % 1 Hz bandwidth
  dW=intlength/subdivisions; % sampling freq =.25 HZ
  W=[];
  W1=[];
  weight=[];
  lowwer=owref(1)-.5*odlength;
  uppper=owref(1)+.5*odlength;
  W=[W lowwer+dW:dW:uppper-dW];
  W1=[W1 lowwer+dW:dW:uppper-dW];
  weight=[weight ones(size(lowwer+dW:dW:uppper-dW))];

  for index=1:loopindex
    lowwer=omegax(index)-.5*intlength;
    uppper=omegax(index)+.5*intlength;
    W=[W lowwer+dW:dW:uppper-dW];
    W1=[W1 lowwer+dW:dW:uppper-dW];
    weight=[weight ones(size(lowwer+dW:dW:uppper-dW))];
   lower=antiresfreq(index)-.5*intlength;
   upper=antiresfreq(index)+.5*intlength;
   W=[W lowwer+dW:dW:uppper-dW];
   W1=[W1 lowwer+dW:dW:uppper-dW];
   weight=[weight ones(size(lowwer+dW:dW:uppper-dW))];
  end

%%compute integrals%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
 DZ_R=[];
 DZ_I=[];
 if meters
   waitbar_handle=waitbar(0,'Computing DZ');
 else
   disp('Computing DZ');
 end
 load stat
 for count=1:length(W)
   if meters
     waitbar(count/length(W));
   end
  z_anal_red=kstat/(j*W(count))+cstat/(j*W(count))+j*W(count)*mstat;
   h_anal_red=inv(z_anal_red);
   hacc=h_anal_red(cset_rel,cset_rel);

   z_exp=k_exp/(j*W(count))+c_exp/(j*W(count))+j*W(count)*m_exp;
   h_exp=inv(z_exp);
   h_exp=h_exp(aset,aset);
   hxcc=h_exp(cset_rel,cset_rel);

   dz=inv(inv(inv(hacc)*(hacc-hxcc)*inv(hacc))-hacc);
   dz_r=real(dz);
   dz_i=imag(dz);
   DZ_R=[DZ_R dz_r(:)];
   DZ_I=[DZ_I dz_i(:)];
 end

 if meters
   close(waitbar_handle)
 end
  W1=1./W1;
 [INTK, INTM, INTC]=intsub(DZ_I,DZ_R,W,W1,weight,cset_size);
 intcorrha=[];
 clear hacc hxcc h_exp z_exp z_anal_red h_anal_red dz
 clear temp ODZ


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
 load stat;
 odcorrha=[];

 kstat(cset_rel,cset_rel)=kstat(cset_rel,cset_rel)+INTK;
 mstat(cset_rel,cset_rel)=mstat(cset_rel,cset_rel)+INTM;
 cstat(cset_rel,cset_rel)=cstat(cset_rel,cset_rel)+INTC;

 if meters
   waitbar_handle=waitbar(0,'Computing Corrected FRF');
 else
   disp('Getting DK/DM/DC')
 end

 for i=1:length(owref)
   if meters
     waitbar(i/length(owref));
   end
   tempza=kstat+j*owref(i)*cstat-owref(i)^2*mstat;
   tempha=inv(tempza);
   odcorrha=[odcorrha tempha(:)];
```

```
end

if meters
  close(waitbar_handle)
end

if loopindex == 1
  lodcorrha1=odcorrha;
  save lod1 lodcorrha1
  clear lodcorrha1
elseif loopindex == 2
  lodcorrha2=odcorrha;
  save lod2 lodcorrha2
  clear lodcorrha2
elseif loopindex == 3
  lodcorrha3=odcorrha;
  save lod3 lodcorrha3
  clear lodcorrha3
elseif loopindex == 4
  lodcorrha4=odcorrha;
  save lod4 lodcorrha4
  clear lodcorrha4
elseif loopindex == 5
  lodcorrha5=odcorrha;
  save lod5 lodcorrha5
  clear lodcorrha5
end


clear tempha tempza kcorrected mcorrected ccorrected
clear INTM INTC INTK temp kstat mstat cstat
load uncorrha
load H_T_EXP

if complete
  plotwhichcoord=[cset(mid_index)];
else
  plotwhichcoord=[cset(mid_index)];
end

for plotindex=1:length(plotwhichcoord)
  which_coord=plotwhichcoord(plotindex);
  h=figure(h+1);
  subplot(2,1,1)
  plot(owref/(2*pi),...
    log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
    which_coord,which_coord,':')))),'r-',...
  owref/(2*pi),...
    log10(abs(odcorrha(ndx3d([aset_size aset_size length(owref)],...
    which_coord,which_coord,':')))),'g--',...
  owref/(2*pi),...
    log10(abs(uncorrha(ndx3d([aset_size aset_size length(owref)],...
    which_coord,which_coord,':')))),'b-.')
  ylabel(['H(',int2str(aset(which_coord)),...
    ',',int2str(aset(which_coord)),') in/lbf (log10 of)'])
  hold on
  v=axis;
  %xlabel('Omega (Hz)')
  if titles
    if use_antires == 'on '
```

```
      title([int2str(odivisions),' PT INTEGRAL SOLTNS WITH ANTIRESONANCES WEIGHTED LEFT TO RIGHT'])
    else
      title([int2str(odivisions),' PT INTEGRAL SOLTNS WEIGHTED LEFT TO RIGHT'])
    end
  end
  plot(omegax(1:loopindex)/(2*pi),...
    ones(1,loopindex)*(v(3)+abs(v(4)-v(3))*.98),'k+')
%   if use_antires
%     %plot(antiresfreq(which_coord,1:loopindex)/(2*pi),ones(1,loopindex)*(v(%3)+abs(v(4)-v(3))*.98),'k*')
%   end
  grid on
  hh=legend('Experimental FRF','Corrected INT Anal FRF','Uncorrected Anal FRF','Included Modes');
  axes(hh)
  hold off
  %if pswitch=='y'
  %  prtfig1(fignum)
  %  delete(h)
  %else
  %  delete(h)
  %end
  subplot(2,1,2)
  thetop=find(abs(owref-W(length(W))*1.3*ones(1,length(owref)))==min(abs(owref-
W(length(W))*1.3*ones(1,length(owref)))));
 % h=figure(h+1);
 % fignum=fignum+1  ;
  plot(owref(1:thetop)/(2*pi),...
    log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
    which_coord,which_coord,1:thetop)))),'r-',...
  owref(1:thetop)/(2*pi),...
    log10(abs(odcorrha(ndx3d([aset_size aset_size length(owref)],...
    which_coord,which_coord,1:thetop)))),'g--',...
  owref(1:thetop)/(2*pi),...
    log10(abs(uncorrha(ndx3d([aset_size aset_size length(owref)],...
    which_coord,which_coord,1:thetop)))),'b-.')
  ylabel(['H(',int2str(aset(which_coord)),',',int2str(aset(which_coord)),')  in/lbf (log10 of)'])
  hold on
  v=axis;
  xlabel('Omega (Hz)')
  if titles
    if use_antires == 'on '
      title([int2str(odivisions),' PT INTEGRAL SOLTNS WITH ANTIRESONANCES WEIGHTED LEFT TO RIGHT'])
    else
      title([int2str(odivisions),' PT INTEGRAL SOLTNS WEIGHTED LEFT TO RIGHT'])
    end
  end
  plot(omegax(1:loopindex)/(2*pi),ones(1,loopindex)*(v(3)+abs(v(4)-v(3))*.98),'k+')
%   if use_antires
%     %plot(antiresfreq(which_coord,1:loopindex)/(2*pi),ones(1,loopindex)*(v(%3)+abs(v(4)-v(3))*.98),'k*')
%   end
  grid on
 %hh=legend('Experimental FRF','Corrected INT Anal %RF','Uncorrected Anal FRF','Included Modes');
 %axes(hh)

  if loopindex == 2
    print -dcdjcolor
    print -dmfile fig6_3
    h=figure(h+1);
    hold off
    load od2
    plot(owref(1:thetop)/(2*pi),...
```

```
        log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'r-',...
        owref(1:thetop)/(2*pi),...
        log10(abs(odcorrha(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'g--',...
        owref(1:thetop)/(2*pi),...
      log10(abs(odcorrha2(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'b-.',...
      owref(1:thetop)/(2*pi),...
        log10(abs(uncorrha(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'m:')
      ylabel(['H(',int2str(aset(which_coord)),',',int2str(aset(which_coord)),')  in/lbf (log10 of)'])
      hold on
      v=axis;
      xlabel('Omega (Hz)')
      if titles
        if use_antires == 'on '
          title([int2str(odivisions),' PT INTEGRAL SOLTNS WITH ANTIRESONANCES WEIGHTED LEFT TO RIGHT'])
        else
          title([int2str(odivisions),' PT INTEGRAL SOLTNS WEIGHTED LEFT TO RIGHT'])
        end
      end
      plot(omegax(1:loopindex)/(2*pi),ones(1,loopindex)*(v(3)+abs(v(4)-v(3))*.98),'k+')
%       if use_antires
%         %plot(antiresfreq(which_coord,1:loopindex)/(2*pi),ones(1,loopindex)*(v(%3)+abs(v(4)-v(3))*.98),'k*')
%       end
      grid on
      hh=legend('Experimental FRF','Corrected INT Anal FRF','Corrected MAT Anal FRF','Uncorrected Anal FRF','Included
Modes');
      axes(hh)
      hold off
      print -dcdjcolor
      print -dmfile fig6_5
    elseif loopindex == 3
      if odivisions == 1
        print -dcdjcolor
        print -dmfile fig6_7
        h=figure(h+1);
      hold off
      load od3
      plot(owref(1:thetop)/(2*pi),...
        log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'r-',...
        owref(1:thetop)/(2*pi),...
        log10(abs(odcorrha(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'g--',...
        owref(1:thetop)/(2*pi),...
      log10(abs(odcorrha3(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'b-.',...
      owref(1:thetop)/(2*pi),...
        log10(abs(uncorrha(ndx3d([aset_size aset_size length(owref)],...
        which_coord,which_coord,1:thetop)))),'m:')
      ylabel(['H(',int2str(aset(which_coord)),',',int2str(aset(which_coord)),')  in/lbf (log10 of)'])
      hold on
      v=axis;
      xlabel('Omega (Hz)')
      if titles
        if use_antires == 'on '
          title([int2str(odivisions),' PT INTEGRAL SOLTNS WITH ANTIRESONANCES WEIGHTED LEFT TO RIGHT'])
        else
```

```
            title([int2str(odivisions),' PT INTEGRAL SOLTNS WEIGHTED LEFT TO RIGHT'])
          end
        end
        plot(omegax(1:loopindex)/(2*pi),ones(1,loopindex)*(v(3)+abs(v(4)-v(3))*.98),'k+')
%         if use_antires
%           %plot(antiresfreq(which_coord,1:loopindex)/(2*pi),ones(1,loopindex)*(v(%3)+abs(v(4)-v(3))*.98),'k*')
%         end
        grid on
        hh=legend('Experimental FRF','Corrected INT Anal FRF','Uncorrected Anal FRF','Included Modes');
        axes(hh)
        hold off
        print -dcdjcolor
        print -dmfile fig6_8

        end
      elseif loopindex == 4
        print -dcdjcolor
        print -dmfile fig6_4
      end
%     if pswitch=='y'
%       prtfig1(fignum)
%       delete(h)
%     else
%       delete(h)
%     end
  end
  delete(h)
  clear H_T_EXP
  clear odcorrha
  clear uncorrha
end

load H_T_EXP
load od1
load od2
load od3
load od4
load uncorrha
%load lod5
    h=figure(h+1);
    plot(owref/(2*pi),...
      log10(abs(H_T_EXP(ndx3d([aset_size aset_size length(owref)],...
      which_coord,which_coord,':')))),'r-',...
    owref/(2*pi),...
      log10(abs(odcorrha1(ndx3d([aset_size aset_size length(owref)],...
      which_coord,which_coord,':')))),'g--',...
    owref/(2*pi),...
      log10(abs(odcorrha2(ndx3d([aset_size aset_size length(owref)],...
      which_coord,which_coord,':')))),'b-.',...
    owref/(2*pi),...
      log10(abs(odcorrha3(ndx3d([aset_size aset_size length(owref)],...
      which_coord,which_coord,':')))),'m--',...
    owref/(2*pi),...
      log10(abs(odcorrha4(ndx3d([aset_size aset_size length(owref)],...
      which_coord,which_coord,':')))),'c-.',...
    owref/(2*pi),...
      log10(abs(uncorrha(ndx3d([aset_size aset_size length(owref)],...
      which_coord,which_coord,':')))),'k-.')
    ylabel(['H(',int2str(aset(which_coord)),...
      ',',int2str(aset(which_coord)),') in/lbf (log10 of)'])
```

```
    hold on
    v=axis;
    %xlabel('Omega (Hz)')
    if titles
      if use_antires == 'on '
        title([int2str(odivisions),' PT INTEGRAL SOLTNS WITH ANTIRESONANCES WEIGHTED LEFT TO RIGHT'])
      else
        title([int2str(odivisions),' PT INTEGRAL SOLTNS WEIGHTED LEFT TO RIGHT'])
      end
    end
   % plot(omegax(1:loopindex)/(2*pi),...
   %    ones(1,loopindex)*(v(3)+abs(v(4)-v(3))*.98),'k+')
   % if use_antires
   %    %plot(antiresfreq(which_coord,1:loopindex)/(2*pi),ones(1,loopindex)*(v(%3)+abs(v(4)-v(3))*.98),'k*')
%     end
    v=axis;
    axis([10 300 v(3) v(4)]);
    grid on
    hh=legend('Experimental FRF','1 mode MAT solution','2 mode MAT solution','3 mode MAT solution','4 mode MAT
solution','Uncorrected Anal FRF');
    axes(hh)
    hold off
    %if pswitch=='y'
    %   prtfig1(fignum)
    %   delete(h)
    %else
    %   delete(h)
    %end


    print -dcdjcolor
    print -dmfile fig6_12
```

SETUP.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% PROGRAM: SETUP.M                                  %
% INITILIZES DATA FILE FOR A FINITE ELEMENT ANALYSIS          %
% VARIABLES SAVED TO FILE SETUP.MAT                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

clc
clear
L=input('Enter the total length of the beam: ');
numel=input('Num of elements: ');
dof_node=input('num dof per node: ');
area=input('area of beam: ');
eeii=input('EI of beam: ');
%ee=input('modulus for beam: ');
pho=input('mass density for beam: ');
numdof=dof_node*(numel+1)
clc
while 1
   conforce=input('Enter # of concentrated loads: ');
   if ~isnan(conforce)
     break
   end
end
for i=1:conforce
   forcepos(i)=input(['Beam position of force ',num2str(i),' : ']);
   forcesiz(i)=input(['Magnitude of force ',num2str(i),' : ']);
end
force=zeros(numdof,1);
for i=1:conforce
   temppos=ceil(forcepos(i)/(L/numel));
   force(dof_node*temppos+1)=force(dof_node*temppos+1)+.5*forcesiz(i);
   force(dof_node*temppos+3)=force(dof_node*temppos+3)+.5*forcesiz(i);

end
while 1
   lmass=input('Enter # of Lumped masses: ');
   if ~isnan(lmass)
     break
   end
end
for i=1:lmass
   masspos(i)=input(['Beam position of Mass ',num2str(i),' : ']);
   masssiz(i)=input(['Magnitude of Mass ',num2str(i),' : ']);
end
lumpmass=zeros(numel,1);
for i=1:lmass
   temppos=round(masspos(i)/(L/numel));
   lumpmass(temppos)=lumpmass(temppos)+masssiz(i);
end
lumpmass'
while 1
   lspring=input('Enter # of Lumped Springs: ');
   if ~isnan(lspring)
     break
   end
end
for i=1:lspring
   springpos(i)=input(['Beam position of Spring ',num2str(i),' : ']);
```

```
    springsiz(i)=input(['Spring constant of Spring ',num2str(i),' : ']);
  end
  lumpspring=zeros(numel,1);
  for i=1:lspring
    temppos=round(springpos(i)/(L/numel));
    lumpspring(temppos)=lumpspring(temppos)+springsiz(i);
  end
  lumpspring'
  conn=[1,2];
  for i=2:numel
    conn=[conn;i,i+1];
  end
  conn
  while 1
    bc=['pinned-pinned    '
       'clamp-clamp      '
       'left guided clamp '
       'right guided clamp'
       'cantilevered     '
       'free-free        '];
    clc
    help bctext
    n=input('Select a boundary condition: ');
    if ((n > 0) & (n < 7))
      break
    end
  end
  if n == 1
    bc='pp';
  elseif n == 2
    bc='cc';
  elseif n == 3
    bc='lc';
  elseif n == 4
    bc='rc';
  elseif n==5
    bc='cl'
  else
    bc='ff';
  end

  clear i
  clear temppos
  clear conforce
  clear forcepos
  clear forcesiz
  clear lmass
  clear masspos
  clear masssiz
  clear lspring
  clear springpos
  clear springsiz
  save setup.mat
  x=0:L/numel:L;
  for i=1:length(x)
   h(i)=5;
  end
  clg
  hold off
  %axis('off')
```

```
%plot([],[])
%hold on
plot(x,h,x,h,'x')
```

BEAMMDL.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%FORM FE BEAM MODEL WITH ERRORS AT  PRESCRIBED LOCATIONS%%%%%%%%%%%
%MASS ERROR LOCATION SPECIFIED VIA POSOFMASSER. VALUE OF ERROR IN%%
%PERCENT GIVEN BY VALOFMASSERR, ETC.%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
clear;clc;clg
casename='Spatially Incomplete A set Stiffness error';
if exist('setup.mat')
  load setup           %load existing input data file
else
  setup              %create new input data file
end
struc_damping=0.00000000001;
lumpdamp=lumpspring*0        ;
posofMasserr=[3 4]  ;
valueofMasserr=[0.25 0.25]   ;
posofStifferr=[3 4];
valueofStifferr=[0.25 0.25]  ;
posofDamperr=[3 4];
valueofDamperr=[0.25 0.25]   ;
for i=1:length(posofMasserr)
  lumpmass(posofMasserr(i))=valueofMasserr(i);
end
for i=1:length(posofStifferr)
  lumpspring(posofStifferr(i))=valueofStifferr(i);
end
for i=1:length(posofDamperr)
  lumpdamp(posofDamperr(i))=valueofDamperr(i);
end
elen=L/numel;
if bc=='pp'
  doftokill=2;
elseif bc=='cc'
  doftokill=4;
elseif bc=='ff'
  doftokill=0;
elseif bc=='cl'
  doftokill=2;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BUILD THE ELEMENTAL  MASS AND STIFFNESS MATRICES%%%%%%%
%FOR A 2DOF/NODE "F.E." STRUCTURES.%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ELEMENT STIFFNESS MATRIX%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ke=[12 6*elen -12  6*elen;
  6*elen 4*(elen^2) -6*elen 2*(elen^2);
  -12 -6*elen 12 -6*elen;
    6*elen 2*(elen^2) -6*elen 4*(elen^2)];

ke=(eeii/(elen^3)).*ke;
g=386;
elemke=eeii/(elen^3);

%ELEMENT MASS MATRICE%%%%%%%%%%%%%%%%%%%%%%%%%%%
me=[156     22*elen       54       -13*elen;
  22*elen  4*(elen^2)    13*elen   -3*elen^2;
```

```
    54     13*elen      156     -22*elen;
  -13*elen -3*(elen^2)    -22*elen  4*(elen^2)];

me = (((pho*area)*elen/g)/(420)).*me;
elemme=(((pho*area)*elen/g)/(420));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%ASSEMBLE GOBAL STIFFNESS AND MASS MATRIX FOR A 2 DOF F.E. STRUCTURE%%%%
%BASED ON THE ELEMENTAL MATRIXES KE AND ME. THE STRUCTURE CONSISTS OF%%%
%NUMEL ELEMENTS WITH ELEMENT CONNECTIVITY GIVEN BY MATRIX CONN%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
goblk=zeros(numdof);
goblm=zeros(numdof);
goblc=zeros(numdof);
for i=1:numel
  v=conn(i,1);
  w=conn(i,2);
  goblk(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)=...
  goblk(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)+...
  ke(1:dof_node,1:dof_node);
  goblk(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)=...
  goblk(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)+...
  ke(1:dof_node,dof_node+1:2*dof_node);

  goblk(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)=...
  goblk(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)+...
  ke(dof_node+1:2*dof_node,1:dof_node);
  goblk(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)=...
  goblk(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)+...
  ke(dof_node+1:2*dof_node,dof_node+1:2*dof_node);

  goblm(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)=...
  goblm(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)+...
  me(1:dof_node,1:dof_node);
  goblm(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)=...
  goblm(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)+...
  me(1:dof_node,dof_node+1:2*dof_node);
  goblm(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)=...
  goblm(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)+...
  me(dof_node+1:2*dof_node,1:dof_node);
  goblm(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)=...
  goblm(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)+...
  me(dof_node+1:2*dof_node,dof_node+1:2*dof_node);
end
goblc=sqrt(-1)*struc_damping.*goblk;
goblkx=goblk;
goblcx=goblc;
goblmx=goblm;


for i=1:numel
  v=conn(i,1);
  w=conn(i,2);
  goblkx(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)=...
  goblkx(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)+...
  lumpspring(i).*ke(1:dof_node,1:dof_node);

  goblkx(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)=...
```

```
    goblkx(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)+...
    lumpspring(i).*ke(1:dof_node,dof_node+1:2*dof_node);

    goblkx(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)=...
    goblkx(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)+...
    lumpspring(i).*ke(dof_node+1:2*dof_node,1:dof_node);

    goblkx(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)=...
    goblkx(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)+...
    lumpspring(i).*ke(dof_node+1:2*dof_node,dof_node+1:2*dof_node);

    goblmx(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)=...
    goblmx(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)+...
    lumpmass(i).*me(1:dof_node,1:dof_node);

    goblmx(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)=...
    goblmx(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)+...
    lumpmass(i).*me(1:dof_node,dof_node+1:2*dof_node);

    goblmx(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)=...
    goblmx(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)+...
    lumpmass(i).*me(dof_node+1:2*dof_node,1:dof_node);

    goblmx(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)=...
    goblmx(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)+...
    lumpmass(i).*me(dof_node+1:2*dof_node,dof_node+1:2*dof_node);


    goblcx(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)=...
    goblcx(dof_node*v-1:dof_node*v,dof_node*v-1:dof_node*v)+...
    lumpdamp(i)*sqrt(-1)*struc_damping.*ke(1:dof_node,1:dof_node);

    goblcx(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)=...
    goblcx(dof_node*v-1:dof_node*v,dof_node*w-1:dof_node*w)+...
    lumpdamp(i)*sqrt(-1)*struc_damping.*ke(1:dof_node,dof_node+1:2*dof_node);

    goblcx(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)=...
    goblcx(dof_node*w-1:dof_node*w,dof_node*v-1:dof_node*v)+...
    lumpdamp(i)*sqrt(-1)*struc_damping.*ke(dof_node+1:2*dof_node,1:dof_node);

    goblcx(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)=...
    goblcx(dof_node*w-1:dof_node*w,dof_node*w-1:dof_node*w)+...
    lumpdamp(i)*sqrt(-1)*struc_damping.*ke(dof_node+1:2*dof_node,dof_node+1:2*dof_node);
end


numdof=numdof-doftokill;

[gk,gm,gc,k_anal,m_anal,c_anal]=fixbcs(goblk,goblm,goblc,bc);

[gkx,gmx,gcx,k_exp,m_exp,c_exp]=fixbcs(goblkx,goblmx,goblcx,bc);
save beamdata
```

FSTATIC.M

```
function [kstat,mstat]=fstatic(k,m,oset,aset)
aset_size=length(aset);
kaa=k(aset,aset);
kao=k(aset,oset);
koo=k(oset,oset);
koa=kao';
clear k;
k=[kaa,kao;koa,koo];

maa=m(aset,aset);
mao=m(aset,oset);
moo=m(oset,oset);
moa=mao';
clear m;
m=[maa,mao;moa,moo];


t_static=-koo\koa;
T_static=[eye(aset_size); t_static];

kstat=T_static'*k*T_static;
mstat=T_static'*m*T_static;

end
```

```
%
function [kirs,mirs]=firs_tam(k,m,oset,aset)
%
% this function returns the IRS reduced stiffness
% and mass matrices, given the unreduced couterparts.
% Care must be taken that the aset and oset vectors correspond
% with the existing arrangement of k and m.
% k and m are UNPARTITIONED matrices.
%
aset_size=length(aset);
%
kaa=k(aset,aset);
kao=k(aset,oset);
koo=k(oset,oset);
koa=kao';
clear k;
k=[koo,koa;kao,kaa];
%
maa=m(aset,aset);
mao=m(aset,oset);
moo=m(oset,oset);
moa=mao';
clear m;
m=[moo,moa;mao,maa];
%
t_static=-koo\koa;
T_static = [t_static; eye(aset_size)];
%
kstat=T_static'*k*T_static;
mstat=T_static'*m*T_static;
%
tirs=t_static+inv(koo)*(moa+moo*t_static)*inv(mstat)*kstat;
T_irs=[tirs;eye(aset_size)];
%
kirs=T_irs'*k*T_irs;
mirs=T_irs'*m*T_irs;
%
% end function firs_tam
```

FREQMODE.M

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%this function returns a vector U containing modal%%%%%%%%%%%%%%%
%frequencies (rad/sec)^2 in ascending order along%%%%%%%%%%%%%%%%%%%%
%with associated mode shapes%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%


function [u1,lambda,index]=freqmode(k,m)
[u,lambdat]=eig(m\k);
[lambda,index]=sort(diag(lambdat));
ll=zeros(length(k),length(k));
for i=1:length(k);
  ll(i,i)=lambda(i);
end
lambda=diag(ll);
for j = 1:length(k)
  u1(:,j) = u(:,index(j));
end
```

NDX3D.M

```
function [r,c] = ndx3d(siz,i,j,k)
%NDX3D         Index into 3-D matrix packed in a 2-D matrix.
%        ELEM = NDX3D([M N P],I,J,K) returns the element position ELEM
%        of the (i,j,k) elements of a M-by-N-by-P matrix which is
%        stored in a (M*N)-by-P matrix.  For example, the three M-by-N
%        matrices A1,A2,A3 are packed into a 2-D matrix using
%          A = [A1(:) A2(:) A3(:)];
%        If length(I) is m, LENGTH(J) is n, and LENGTH(K) is p,
%        then ELEM will be an m-by-n-by-p matrix.
%
%        [R,C] = NDX3D([M N P],I,J,K) returns the row and column
%        position of the (i,j,k) element as stored in the normal
%        2-D matrix of size (M*N)-by-P.
%
%        To specify all the elements along one dimension use ':'.
%        For instance, NDX3D([3 5 4],2:3,':',3:4) returns the
%        elements for the 2-by-5-by-2 matrix.
%
%        See also ELEM3D, MESHGRID, SLICE.

%        Clay M. Thompson 11-3-92
%        Copyright (c) 1992 by The MathWorks, Inc.
%        $Revision: 1.7 $  $Date: 1993/09/03 14:36:52 $

if isstr(i), i = 1:siz(1); end
if isstr(j), j = 1:siz(2); end
if isstr(k), k = 1:siz(3); end

if isempty(i) | isempty(j) | isempty(k), r = []; c = []; return, end

if any( (i<0) | (i>siz(1)) ), error('Index I out of range.'); end
if any( (j<0) | (j>siz(2)) ), error('Index J out of range.'); end
if any( (k<0) | (k>siz(3)) ), error('Index K out of range.'); end

if nargout==2
  [jj,ii] = meshgrid(j,i);
  r = ii(:) + (jj(:)-1)*siz(1);
  c = k(:);

else
  [jj,ii,kk] = meshgrid(j,i,k);
  r = ii + (jj-1)*siz(1) + (kk-1)*prod(siz(1:2));

end
```

INTSUB.M

```
function [K,M,C]=intsub(Z_I, Z_R, omega, omega1, W, set_size)
waitbar_handle=waitbar(0,'Computing Integrals');
j=sqrt(-1);

integ=omega.^2.*abs(W);
rint=real(integ);
iint=imag(integ);
aa=mytrapz(omega1,rint)+mytrapz(omega1,iint)*j;
integ=(1./omega.^2).*abs(W);
rint=real(integ);
iint=imag(integ);
bb=mytrapz(omega1,rint)+mytrapz(omega1,iint)*j;
integ=abs(W);
rint=real(integ);
iint=imag(integ);
cc=mytrapz(omega1,rint)+mytrapz(omega1,iint)*j;
for i=1:set_size
 waitbar(i/set_size);
 for k=i:set_size
  M(i,k)=(1/(aa*bb-cc^2))*(bb*mytrapz(omega1,...
   omega.*Z_I(ndx3d([set_size set_size length(omega)],...
   i,k,1:length(omega))).*abs(W))-cc*mytrapz(omega1,...
   1./omega.*Z_I(ndx3d([set_size set_size length(omega)]...
   ,i,k,1:length(omega))).*abs(W))) ;
  M(k,i)=M(i,k);
  C(i,k)=(1/cc)*mytrapz(omega1,Z_R(...
   ndx3d([set_size set_size length(omega)],...
   i,k,1:length(omega))).*abs(W));
  C(k,i)=C(i,k);
  K(i,k)=(1/(aa*bb-cc^2))*(cc*mytrapz(omega1,...
   omega.*Z_I(ndx3d([set_size set_size length(omega)]...
   ,i,k,1:length(omega))).*abs(W))-aa*mytrapz(omega1,...
   1./omega.*Z_I(ndx3d([set_size set_size length(omega)]...
   ,i,k,1:length(omega))).*abs(W))) ;
  K(k,i)=K(i,k);
 end
end
close(waitbar_handle)
```

MYTRAPZ.M

```
function z = trapz(x,y)
%TRAPZ  Trapezoidal numerical integration.
%       Z = TRAPZ(X,Y) computes the integral of Y with respect to X using
%       trapezoidal integration.  X and Y must be vectors of the same length,
%       or X must be a column vector and Y a matrix with as many rows as X.
%       TRAPZ computes the integral of each column of Y separately.
%       The resulting Z is a scalar or a row vector.
%
%       Z = TRAPZ(Y) computes the trapezoidal integral of Y assuming unit
%       spacing between the data points.  To compute the integral for
%       spacing different from one, multiply Z by the spacing increment.
%
%       See also SUM, CUMSUM.

%       Clay M. Thompson, 10/16/90; Cleve Moler, 1/19/92.
%       Copyright (c) 1984-94 by The MathWorks, Inc.

%  Make sure x and y are column vectors, or y is a matrix.

%  Trapezoid sum computed with vector-matrix multiply.
[m,n]=size(x)  ;
z=0          ;
for index=1:m
  yy=y(1+(index-1)*n:n+(index-1)*n);
  yy=yy(:);
  xx=x(index,:);
  xx = xx(:);
  z =z+diff(xx)' *(yy(1:n-1) + yy(2:n))/2;
end
```

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Professor J.H. Gordis, Code ME/GO Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5002	2
4.	Department Chairman, Code ME Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5002	1
5.	Naval Engineering Curricular Office (Code 34) Naval Postgraduate School Monterey, CA 93943-5002	1
6.	LCDR. Richard Johnson 1474 15th Street Imperial Beach, CA 91932	2